# Exhibit A

# of

# 37 C.F.R. § 1.131 Declaration

# of Michael G. West

# MERLIN

# DIGITAL DISPLAY CONTROLLER

# ARCHITECTURE SPECIFICATION

## *FINAL DRAFT*

IN FOCUS SYSTEMS, INC.

# TABLE OF CONTENTS

# DEFINITIONS

| | |
|---|---|
| **ASIC** | Application Specific Integrated Circuit (30-40K gate gatearray in this context). |
| **DVB** | Infocus Systems Digital Video Bus. |
| **Field** | For interlaced signals one half of the entire image. For non-interlaced signals fields and frames are equivalent and represent a single scan of the entire image. |
| **Field Memory** | Memory for storage of one entire interlaced field (i.e., 1/2 of a frame memory). |
| **Frame** | A single scan of the image. For interlaced signals one frame equals two fields (odd & even). |
| **Frame Memory** | Memory for storage of one entire frame of video. |
| **FRM** | Frame Rate Modulation, a technique for producing intermediate gray levels on an LCD display. |
| **High Resolution** | For the purposes of this document High Resolution implies a 800x600 or larger image size. |
| **HSync** | Horizontal sync pulse used to indicate the start of a line. |
| **Lorikeet** | An embedded Adobe™ PDF player engine. |
| **Low Resolution** | For the purposes of this document Low Resolution implies a 640x480 or less image size. |
| **MAC** | Merlin Address and Control Block. |
| **Merlin** | The architecture defined in this document. |
| **MID** | Merlin Input Data Path Block. |
| **MOD** | Merlin Output Data Path Block. |
| **RAM** | Random Access Memory. |
| **SAM** | Serial access memory commonly found in VRAM's and WRAM's. |
| **Scribble Mode** | The ability to draw on top of the image with a pointing device. |
| **SOG** | Sync on Green, a composite sync signal imposed on the green analog video channel. |
| **TFT LCD** | Thin-Film-Transistor based LCD display capable of video rate refresh. |
| **UFP** | Ultra-Fast Page Mode WRAM operations. |
| **VRAM** | Video RAM: A common dual ported memory component with a dynamic RAM based memory cell structure. |
| **VSync** | Vertical sync pulse used to indicate the start of a video frame. |
| **WRAM** | Window RAM: A Samsung 8 Mbit dual ported memory device. |

# 1. INTRODUCTION

Competitive trends in the projection display industry, along with customer feedback on ease-of-use requirements indicates that a major revision of the state of the art for digital display control is required. The project to develop this system is called Merlin. This document describes the architecture of the system, along with the ASIC necessary to implement it. To push the limits of ease-of-use three key issues will be addressed: Hands free operation, Image Artifact Reduction, and Compatibility.

The first issue, hands free operation can best be described as the unit's ability to recognize the connected video source and optimally configure the electronics for operation. Key features enabling this are: AutoSync, Auto Position, Auto Tracking, Auto White & Black Level Detection, and Front End image processing to reduce overall system noise. Merlin will in some cases resolve these issues through direct hardware support and in others will provide the hardware necessary for sophisticated software algorithm's to implement the solution.

The next issue, image artifacts, has two primary causes: noise and frame rate modulation, both of which exist in the current generation of products. The noise is mainly introduced by the analog front end and the digitization process. The second cause frame rate modulation is the result of the technology utilized to produce more colors on a display than are naturally possible. This technology produces the illusion of more colors by averaging over time the intensity of a pixel thereby depending on the eye for integration. This results in the appearance of noise in the image. To make matters worse when these effects are combined there is a multiplicative effect which causes a significant increase in the perceived artifacts. Merlin will utilize improved signal processing techniques (hysteresis) and new programmable FRM algorithms to minimize these artifacts. In addition as TFT-LCD technology migrates to a higher number of natural colors the overall FRM effects will be greatly reduced.

Finally the issue of compatibility relates to the products ability to tolerate a wide range of video sources. The issues here range from resolution and resultant pixel rate to separate versus composite sync sources. Examples of where current product falls short are: No compatibility with high resolution input through image resizing, Positive polarity composite sync signals are non-compatible, and some relationships between VSync and HSync are illegal. Merlin will address these issues and many more relating to our products compatibility.

In addition to the ease-of-use and performance issues, a significant trend in the memory industry away from the VRAM's requires a completely new approach to frame memory design. As a result Merlin will be the first architecture to utilize the newly emerging WRAM memory technology which can implement a 640x480 24 bit color frame memory in a single IC!

Aside from performance issues related to ease-of-use Merlin will address performance in many other areas including: image resizing, color resolution, menu system look & feel, video windowing (future), and 3 through 8 bit TFT support. Refer to the list on the following page for a complete system level performance related feature set.

Finally a primary goal of Merlin is to incorporate a significantly enhanced feature set into the product line with little impact on the cost of the electronics.

## 1.1 KEY FEATURES:

**Modular Architecture.**
- *Two Video Card Channels.*
- *One Processor Card Channel.*

**Adobe Friendly.**
- *32 Bit Direct Frame Memory Interface.*
- *Fast Posted Frame Memory Writes.*
- *Adobe Video Overlay's (Full 24bit color).*

**Robust Video Interface.**
- *Composite Sync Stripper.*
- *Sync polarity detect.*
- *Highly Accurate Mode Detection Timers.*

**Auto Everything.**
- *Auto Position.*
- *Auto Sync (Automatic clock phase adjust).*
- *Auto Black & White Level Detect with Channel Balancing.*
- *Auto Tracking.*
- *Automatic Sync Source Detect.*

**16.7 Million Colors.**
- *24bit's per pixel.*
- *Programmable Frame Rate Modulation (FRM).*
- *Programmable Dithering.*

**Video Windowing**
- *Programmable Picture in Picture.*

**640x480 & 800x600 LCD Support.**
- *3,4,6, and 8 Bit TFT Support.*

**Screen Overlays.**
- *"Scribble mode".*
- *On Screen Pointer.*
- *Menus: Any Size, Shape or Number.*

**High Resolution Resizing (External).**
- *High Input Bandwidth, with 1/2 freq sample option.*
- *Genesis Compatible.*

**Programmable Gamma Correction.**
- *Full Three Channel Color Lookup Table (CLUT).*

**Digital Noise Reduction.**
- *Programmable Input Hysteresis.*

**Highly Integrated System.**
- *68K Based Micro Interface.*
- *High Speed WRAM Controller.*

# 2. ARCHITECTURE OVERVIEW

Figure 2-1 illustrates Merlin's basic partitioning and system architecture. At the core of this architecture is a frame memory based on the WRAM. The frame memory based architecture is required to support a wide range of display technologies and to implement numerous features such as: wide compatibility, high-resolution resizing, rear projection (on some LCD's), ceiling mount (Assuming projector is upside down), and scribble mode. If specific display types were targeted and some reduction in compatibility tolerated (i.e., fast frame rate SVGA modes) then a totally FIFO based architecture could be used resulting in a significant complexity/cost savings. Unfortunately this approach only works well for 640x480 display modes and breaks down when higher resolutions are supported through resizing and/or high-res display devices.

Surrounding the WRAM frame memory is the Merlin ASIC which is responsible for implementing the heart of the Merlin system. This ASIC is partitioned into three blocks: an Input Data Path (MID) block which buffers and packs data for transfer to the WRAM, an Output Data Path (MOD) block which unpacks and buffers data from the WRAM for transfer to the LCD and a WRAM Address & Control (MAC) block which controls where data is read and written in the frame memory. Combined these chips implement all of the video processing, LCD & WRAM control, and ease of use features of the Merlin architecture.

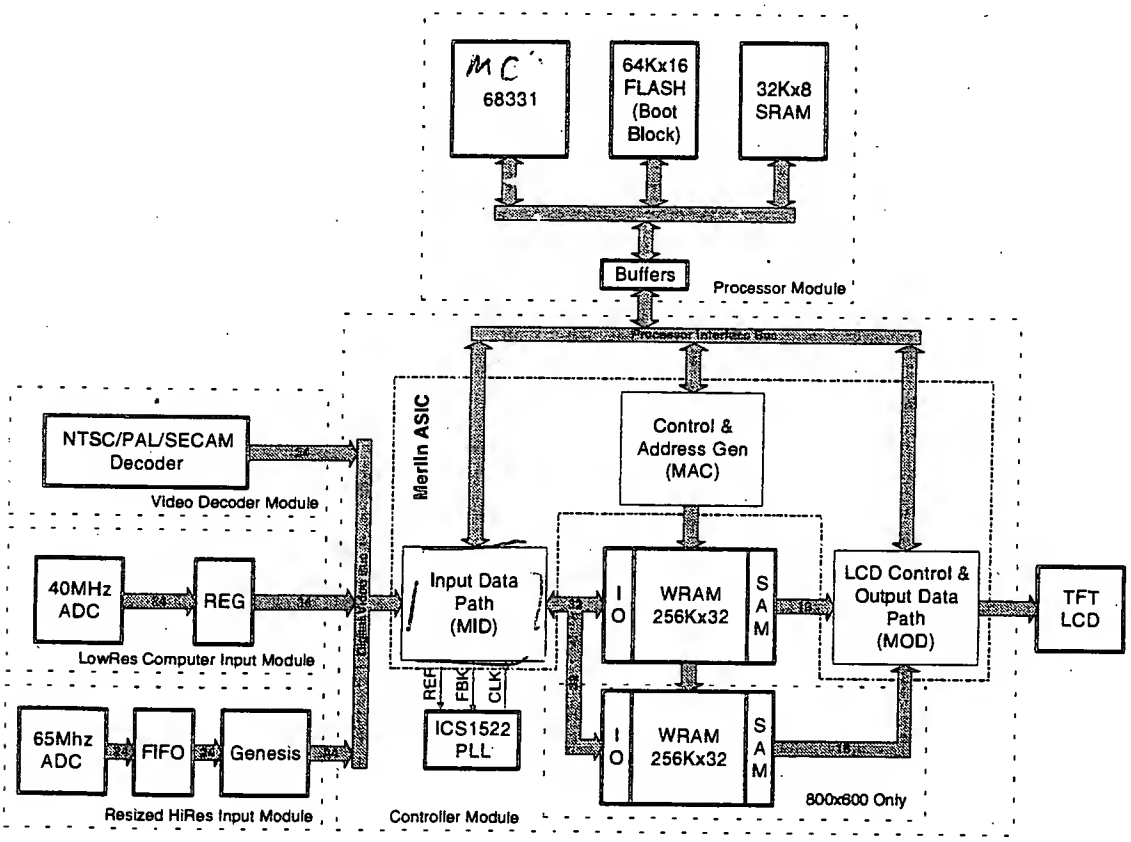



**Figure 2-1: Merlin top level architecture block diagram.**

The Merlin core is tied to the external system via two buses. The first, on the input side, is a digital video bus (DVB) which is designed to allow multiple sources with on the fly source select controlled by the MAC Block. The second is a processor interface bus (PIB) which is used for

system control and on-screen-display. This architecture allows for a modular design where the capabilities and performance of the modules can change without impacting the Merlin core chipset. For instance in the base line system the processor module consists of 68331 based embedded controller which has 68020 class performance, however, for presentation player systems this module can be replaced by a 68EC040 class card which allows for high speed rendering of graphical data.

## 2.1 Modularity

As was mentioned before Merlin is a modular architecture which supports two kinds of modules: one for the processor which controls the system, and the other for video capture. The electrical interfaces and firmware visible features of this modularity are described in the following pages. However, a key point is that Merlin is not intended to be an auto configuring or "hot plugable" system. There will be a primitive card ID feature, however, no provisions will be made for automatic firmware upgrade. If a new module is installed and the firmware recognizes the ID then the system will operate with no further action required, otherwise, new firmware which understands the communications protocols of the new module will need to be installed into FLASH ROM via the serial port. This form of modularity supports dealer upgradability fairly well, however, for end users this is less than ideal. Approaches beyond this would require some form of intelligence or storage of firmware on the modules themselves, which would increase the cost of the system.

## 2.2 The WRAM and Basic System Timing

The WRAM seen in Figure 2-2 is architecturally similar to a VRAM. The main difference is that the WRAM transfers only 256 bits per transfer cycle to the SAM array. This reduced SAM input bandwidth results in a greater overhead requirement to keep the SAM from running out of data. Unlike VRAM's which require one transfer cycle per display row, WRAM's require a total of 64. In order to reduce this overhead Merlin overlaps SAM writes with reads when the SAM is nearly empty. During this time the parallel port is used to initiate transfers to the SAM. A total of eight transfer cycles are required to fill the SAM. The WRAM (Figure 2-2) is a dual ported device containing a total of 8M bits of storage, organized as 512 thirty-two bit wide columns by 512 rows resulting in 262K 32bit words. Since 307K pixels are required for 640x480 images, the 24 bit pixel data must be packed into the 32 bit wide WRAM words. Packing allows 640 pixels to be stored in the 512 columns of each row. A side benefit of packing is a four-thirds increase in bandwidth. The parallel and serial ports of the WRAM are used for video input (parallel) and LCD output (serial). The parallel port is also used for micro-controller frame memory access. For 800x600 resolution systems two WRAM's are used and they are interleaved into odd and even banks. Interleaving is required to support the clock rate of 800x600 LCD's. Also, for 600 line displays more than one line of display data must be packed into a WRAM row.

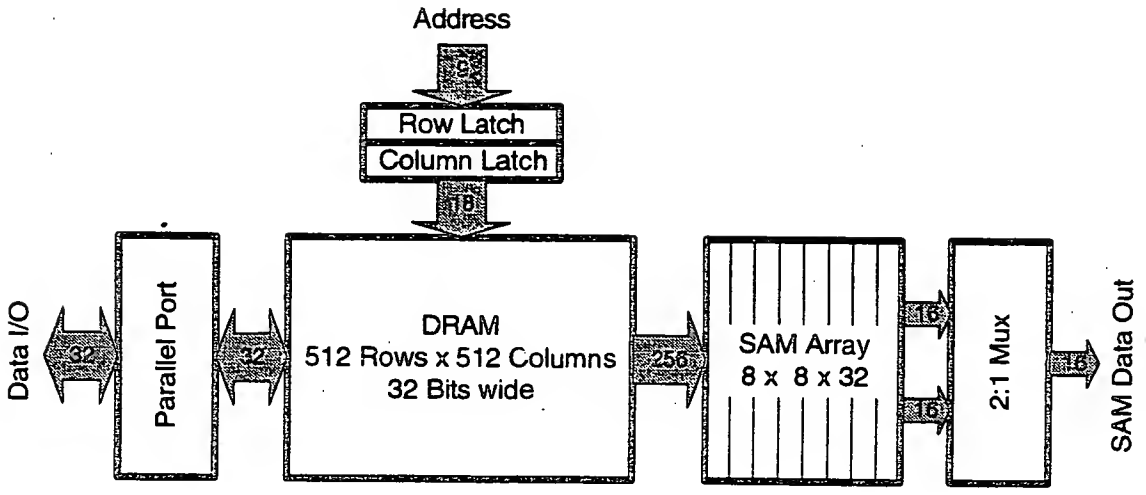



**Figure 2-2: WRAM block diagram.**

Another unique feature of Merlin, illustrated in Figure 2-3, is that input and output FIFO's are used to bridge the asynchronous timing between the video input, WRAM and LCD output. The main advantage of this approach is to totally isolate the LCD output and control from the video clock which is at times unstable. Also, by placing a FIFO between the WRAM and LCD allows the LCD timing to be independent of the WRAM clock rate and can in fact be up to 4/3rd's faster since this is the pixel to word ratio.

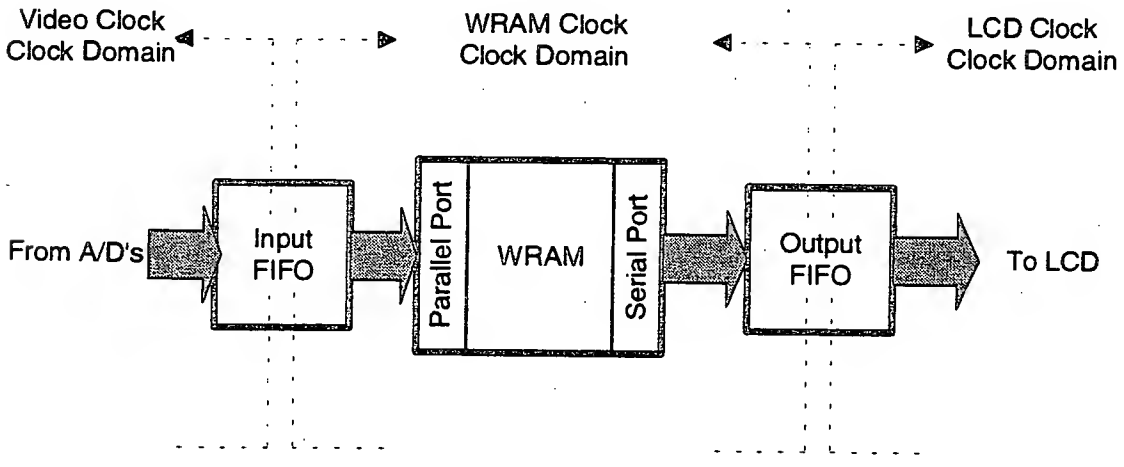



**Figure 2-3: FIFO and Clock Architecture.**

## 2.3 Input and Output Bandwidth

Isolating the WRAM in between two FIFO's allows for maximum bandwidth to be achieved on both the input and output sides and also leads to a simplification of the overall WRAM controller's complexity. On the output side as indicated before the LCD clock can run at 4/3's the output FIFO's input data rate which is one half the 50 Mhz WRAM clock because of the multiplexed SAM. This yields a maximum LCD clock rate shown below (640x480 case):

*Max LCD Clock Rate = 33.3 Mhz = (4/3) * (1/2) * 50 Mhz*

The maximum input bandwidth is more difficult to determine since it is dependent on the LCD clock rate chosen. If a worst case 33 Mhz LCD clock is used then the max input bandwidth can be determined as follows:

*Max Input Bandwidth = 58.3Mpixels/sec = (4/3) * ((Sd -Rt*Ou)/(Sd) * (50Mhz)*

*Where: Sd = SAM depth = 128 (16 bit words)*

*Ou = Output bandwidth utilization = (4/3) * 25/33 = 1*

*Rt = SAM refill time in number of clocks = 16*

## 2.4 Compatibility

Merlin is architected to be the most highly compatible low resolution & medium resolution product in the industry. Key points enabling this are the high input bandwidth (described above) and the ability to support (and in fact facilitate) external resizing engines. The Merlin core can support up to 75Hz refresh rate 1024x768 modes, however, since resizing technology does not currently exist which can operate at these clock rates the high resolution support will be limited to 60Hz 1024x768 modes. In fact any mode with a clock rate less than 65MHz will be supported. This is a vast improvement over the current generation of low cost product. It should be noted that high resolution interlaced modes such as XGA will not achieve good resizing results because in the vertical direction the information is not available to the resizing chip. It will be possible to support these modes through a simple line and pixel dropping algorithim. See Appendix A. for a list of the supported video modes. Note since Merlin is an adaptive technology many modes not on the list will be supported.

## 2.5 Processor Interface

A key feature of Merlin is its Processor Interface Bus. This is the main pathway of communication between the video display and embedded presentation engines such as Adobe. Merlin provides a full 32 bit wide multiplexed address and data bus for this path. This is a high speed interface which can sustain write cycles at a rate of 1/4 the processor clock rate. This implies that for a 25Mhz processor data can be written at a rate of 6.25MWords/Sec (25MBytes/Sec). From this you can derive Table 2-1 which illustrates the time to redraw the entire screen based on the display resolution. Off course this table assumes the processor can actually keep up, an unlikely event even for the 68EC040. As a result do not use these numbers as an indication of how fast a 68331 could draw a menu, they are unrelated. The point is that this is theoretical maximum bandwidth sustainable by Merlin over the processor bus.

| Screen Resolution. | Screen Redraw Time. | 1/4 Area Refresh Rate. |
|---|---|---|
| 640x480 | 54msec | 73Hz |
| 800x600 | 85msec | 47Hz |
| 1024x768 | 139msec | 29Hz |
| 1280x1024 | 231msec | 17Hz |

Merlin accomplishes these performance numbers by using using ultra-fast page mode transfers when consecutive writes are targeted at the same row of the WRAM. This requires that processor writes have priority of image capture writes and that when a processor write occurs all remaining capture writes to the display row are cancelled. Note that the above numbers assume an approximate 10% display refesh overhead. Also, for read cycles the performance is much lower and they should generally be avoided.

## 2.6 Merlin Address Map.

The Merlin address space illustrated in Figure 2-4 consists of a single 4Mbyte block of memory located at a base addresses determined by the processor module. Display memory starts at address DisplayBase+0 and continues until the end of display memory. Control registers are mapped at the end of this block and are illustrated below. The display window has three fundamental modes of operation (described in the following sections), one for 640x480 displays and the other two for 800x600 displays.

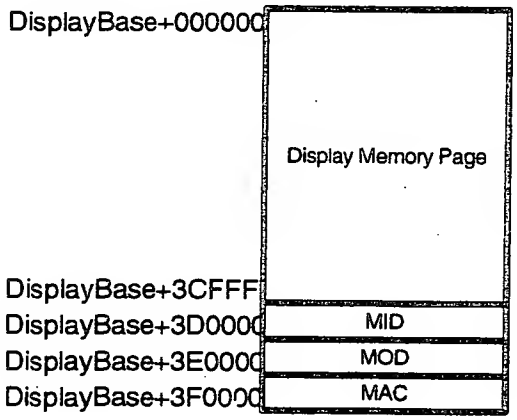



**Figure 2-4: Merlin Memory Map.**

### 2.6.1 640x480 Panel Display Memory Addressing.

A 640x480 24bit display requires just under 1 megabyte of address space which requires 20 bits to address. As was mentioned in the previous section the base address of the display page is determined by firmware. The address architecture shown in Figure 2-5 is based on a 512 word line size resulting in up to 682 pixels per line. Only the first 640 pixels in a line are visible on the display. The main advantage of using a 512 word line size is that WRAM rows and display rows

become equivalent. This helps reduce the transfer cycle overhead caused by display updates yielding the maximum possible input bandwidth.

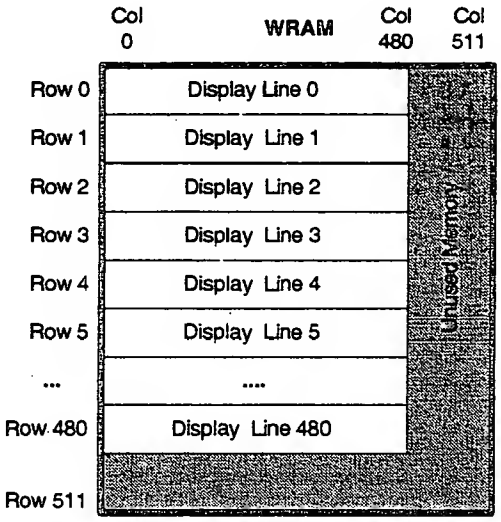



**Figure 2-5: 640x480 Display WRAM Organization (SVGAEn=0).**

The organization of pixels in the WRAM is illustrated in Table 2-2. Note that the 24 bit RGB pixel data is packed into the 32 bit wide WRAM word (Figure Notation: R0,1 = Red data for row 0 column 1).

The byte address of a pixel in display space can be calculated as follows:

*Address = 2048*row + 3*column*

The address of the red, green, and blue components can be computed by adding an offset from 0 to 2 (0 for red, 1 for blue, 2 for green). The following example illustrates how an arbitrary pixel would be accessed.

*If row = 250 and column = 75 then*

*Address = 2048*250 + 3*75 = 512225 = 0x7D0E1*

To access the blue component of this pixel an offset of two would be added to the address resulting in an byte address of 0x7D0E3. For firmware to access this pixel, the display base would be added to this value and a byte memory operation would be issued.

Table 2-2: Video memory address map

| WRAM Row | Base Address | Sub Address 0 | 1 | 2 | 3 |
|---|---|---|---|---|---|
| 0 | 000 | R0,0 | G0,0 | B0,0 | R0,1 |
| 0 | 004 | G0,1 | B0,1 | R0,2 | G0,2 |
| 0 | 008 | B0,2 | R0,3 | G0,3 | B0,3 |
| 0 | 00c | R0,4 | G0,4 | B0,4 | R0,5 |
| ..... | | | | | |
| 0 | 774 | R0,636 | G0,636 | B0,636 | R0,637 |
| 0 | 778 | G0,637 | B0,637 | R0,638 | G0,638 |
| 0 | 77c | B0,638 | R0,639 | G0,639 | B0,639 |
| *unused* | | | | | |
| 1 | 800 | R1,0 | B1,0 | G1,0 | R1,1 |
| ..... | | | | | |

### 2.6.2 800x600 Panel Display Memory Addressing with Full Speed Sample Rate.

Addressing for 800x600 displays is more complicated than the 640x480 case because two WRAM's are required to hold the necessary 480000 Pixels (Over 11.5Mbits of data requiring a 22bit address). Also in order to support the required 600 display lines more than one row of image data must be packed into a WRAM row. To meet these requirements Merlin uses a odd/even bank approach with a line size of 640 words (320 words per bank). This line size was chosen to ensure that all line fragments in a WRAM row were divisible by eight. This simplifies the electronics because of the 8 word SAM transfer bus width. The 320 word line size results in a display mapping which repeats after only 5 WRAM rows (See Figure 2-6) and can support up to 819 display lines, of which only the first 600 are visible. Unfortunately the 320 word line size increases the complexity of determining where a row in display memory starts and in addition yields WRAM rows which contain fragments of multiple display lines. The saving feature which prevents loss of input bandwidth is that with two WRAM's the SAM output bandwidth is doubled resulting in fewer transfer cycles. This is true because the clock rate for 800x600 LCD's is only 60% faster than the 640x480 case.

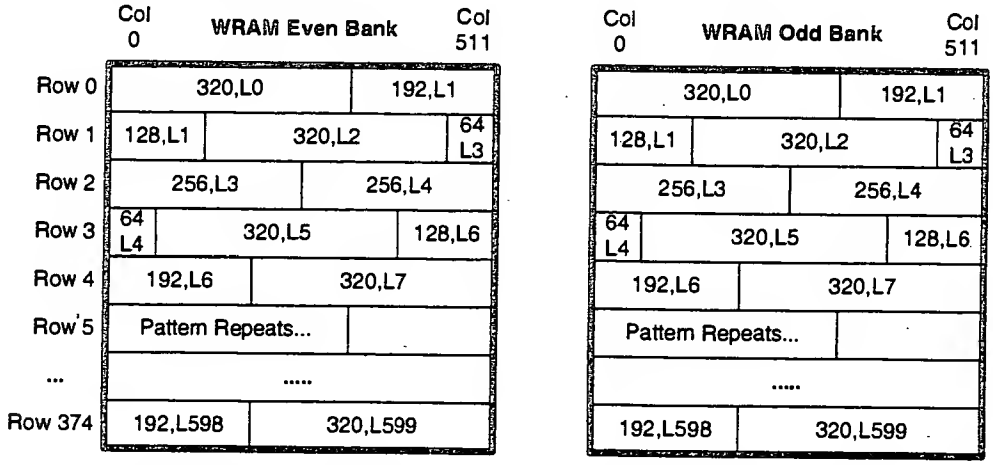



**Figure 2-6: 800x600 Display WRAM Organization (SVGAEn=1).**

From a firmware perspective the following formula can be used to calculate the byte address of a pixel in display memory.

*Address = 2560*row + column*3*

The address of the red, green, and blue components can be computed by adding an offset from 0 to 2 (0 for red, 1 for blue, 2 for green). The following example illustrates how an arbitrary pixel would be accessed (Note: 2560 = 2048+512).

*If row = 250 and column = 75 then*

*Address = 2560*250 + 3*75 = 640225 = 0x9C4E1*

To access the blue component of this pixel an offset of two would be added to the address resulting in an byte address of 0x9C4E3. For firmware to access this pixel, the display base would be added to this value and a byte memory operation would be issued.

## 2.7 800x600 1/2 Frequency Pixel Sampling Memory Addressing.

Like the previous memory mode this mode requires two WRAM's, however, a different pixel grouping in memory is required to allow for the maintanence of maximum bandwidth. The line grouping (Figure 2-6) is the same as standard 800x600 mode, however, the organization of pixels in each WRAM bank is based on placing odd pixels in the odd WRAM bank and even pixels in the even WRAM bank. illustrates the pixel mapping from the PIB intefaces perspective.

Table 2-3 SVGA 1/2 Sample Frame Buffer

| WRAM Bank | Base Address | Sub Address | | | |
|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 |
| Even | 000 | R0,0 | G0,0 | B0,0 | R0,2 |
| Odd | 004 | R0,1 | G0,1 | B0,1 | R0,3 |
| Even | 008 | G0,2 | B0,2 | R0,4 | G0,4 |
| Odd | 00c | G0,3 | B0,3 | R0,5 | G0,5 |
| Even | 010 | B0,4 | R0,6 | G0,6 | B0,6 |
| Odd | 014 | B0,5 | R0,7 | G0,7 | B0,7 |
| ..... | | | | | |

## 2.8 On Screen Menus and Overlays.

Merlin uses the write mask feature of the WRAM to implement screen overlays (menu's & scribble mode) under the control of the micro controller. In order to allow firmware to overwrite the active video image the least significant bit of every byte is write protected via the write mask during video data write cycles. This results in a one bit reduction in color resolution when the overlay feature is enabled. The three protected bits can be written by firmware allowing individual pixel control of the screen overlays. When these three bits are each zero then the 21 bits of digitized video is used for output display, otherwise, the three bits are used as a lookup into a 9 bit wide palette array. This allows for up to seven colors out of a palette of 512 to be used in the on screen display. Since menu's destroy scribble data and there is insufficient processor memory to save this information these to modes of operation are mutually exclusive. This implies that they can use different palettes allowing for colors which are better suited for

hilighting when in scribble mode. Refer to the MOD block diagram in Figure 4-1 for a illustration of the palette and pixel selection hardware. This approach offers a reasonable compromise between cost, complexity and performance with the main drawback being the limited number of colors available for Scribble mode and Menu's.

Another Merlin on-screen display feature is a full 24 bit protected window in the video input. This rectangular region can be used for displaying CPU data in a window while the input data changes around it. This window can be used to display menus in more colors than is possible in overlay mode.

# 3. MERLIN INPUT DATA PATH (MID)

Figure 3-1: Input data path block diagram. illustrates the Merlin Input Data Path (MID) which acts as a buffer between the digital video bus and the WRAM. A 48 pixel deep FIFO is used to absorb momentary interruptions of the WRAM parallel input caused by transfer cycles. This chip also formats and packs the data into 32 bit words for transfer to the WRAM.

The MID Block has two fundamental modes of operation, capture and compare. Capture mode is used to collect the pixel data for display without any noise reduction. In this mode the incoming pixel data is written to the WRAM via the bi-directional memory port. Compare mode is used for digital noise reduction and autosync. In compare mode MID accepts data from the bi-directional memory port and compares it with the incoming pixel data. If the data has changed by more than a threshold value then a counter is incremented and the MissDetect flag is set. The MissDetect flag is returned to the MAC controller forcing the system to switch from compare mode to capture mode on the next frame. Once a frame has been written the controller switches back to compare mode.

Finally MID includes a number of functions necessary for system setup and configuration. These ease of use features include: Image position detection, Black and White level detect, and the prior mentioned compare mode for noise reduction. In addition this chip is responsible for capture control and PLL feedback generation.

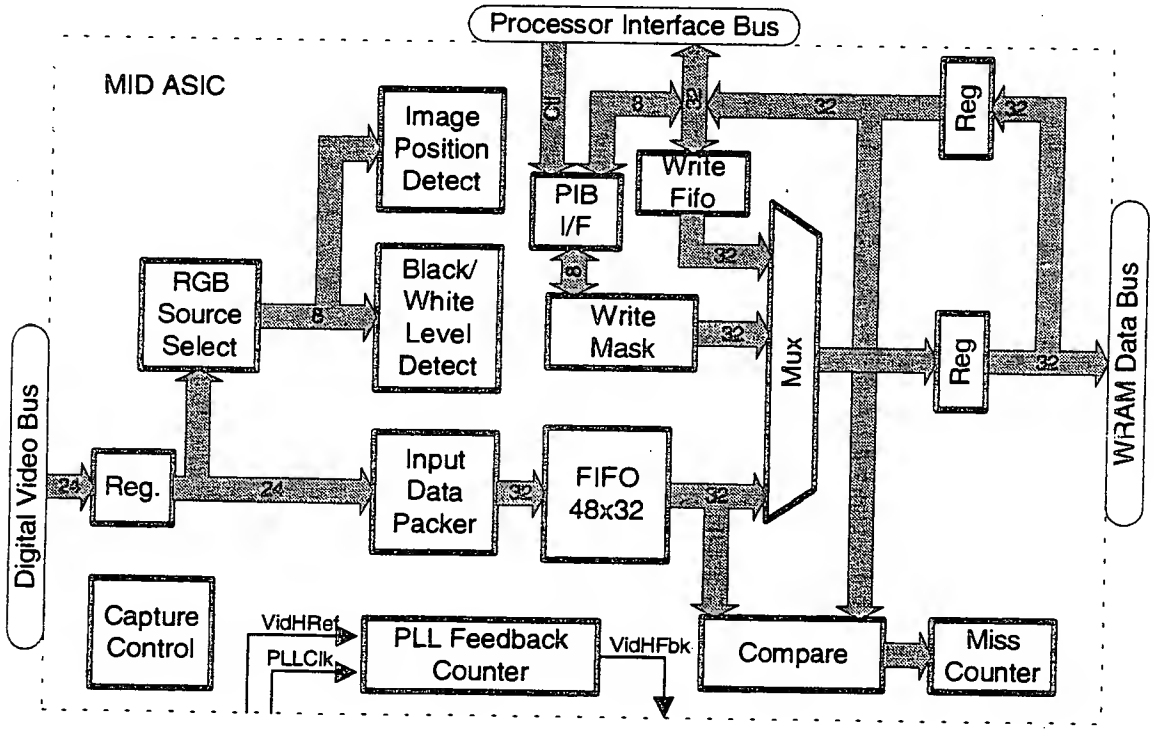



**Figure 3-1: Input data path block diagram.**

## 3.1 MID Register Address Map.

The software visible registers of the MID Block are accessed through a parallel micro-controller interface which is compatible with the processor interface bus (PIB). When these registers are addressed 8-bits are transfered on the least significant byte of the 32 bit processor data bus. Software visible status registers are updated only at the beginning of VSync and a lock semaphore is provided to prevent reading data which is in the process of updating. The Lock bit is contained in the misc. control register in the MAC Block. Table 3-1 contains a list of all software visible registers in the MID Block. Access of these registers is enabled by the uMIDCS chip select which is decoded from the processor address bus by the MAC Block.

Table 3-1 MID Register Map

| Base+3D0000+Offset | | | Bit Position | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Function | r/w | Offset | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ChipRev | r | 03 | ChipRev(7:0) | | | | | | | |
| Clock Control | r/w | 07 | PLLClkSel | IntFbkSel | ClkEn(2:0) | | | PLLPol | RGBSel(1:0) | |
| Write Mask | r/w | 0B | WriteMask(7: 0) | | | | | | | |
| Threshold | r/w | 0F | ThreshHold(7:0) | | | | | | | |
| Misc. Control | r/w | 13 | BlkEdge | AutoEdge | VidWinEn | ExtHRef | | | | OddStart |
| Horiz. Compress | r/w | 17 | HDrop | VAvg | HAvg | HScale(4:0) | | | | |
| Miss Counter | r | 1B | | | | | MissCnt(19:16) | | | |
| | r | 1F | MissCnt(15:8) | | | | | | | |
| | r | 23 | MissCnt(7:0) | | | | | | | |
| Image Top | r | 27 | | | | | | Top(10:8) | | |
| | r | 2B | Top(7:0) | | | | | | | |
| Image Bottom | r | 2F | | | | | | Bottom(10:8) | | |
| | r | 33 | Bottom(7:0) | | | | | | | |
| Image Left | r | 37 | | | | | | Left(10:8) | | |
| | | 3B | Left(7:0) | | | | | | | |
| Image Right | r | 3F | | | | | | Right(10:8) | | |
| | r | 43 | Right(7:0) | | | | | | | |
| Blanking Level | r | 47 | BlankLevel(7:0) | | | | | | | |
| Black Level | r | 4B | BlackLevel(7:0) | | | | | | | |
| White Level | r | 4F | WhiteLevel(7:0) | | | | | | | |
| Clocks per line | r/w | 53 | | | | | | ClksPerLine(10:8) | | |
| (Tracking) | r/w | 57 | ClksPerLine(7:0) | | | | | | | |
| PLL Coast Control | r/w | 5B | CoastEn | CoastStop(3:0) | | | | CoastStart(10:8) | | |
| | r/w | 5F | CoastStart(7:0) | | | | | | | |
| Vertical Position | r/w | 63 | VidActTop(7:0) | | | | | | | |
| Horizontal | r/w | 67 | | | | | | | | VidActLeft(9:8) |
| Position | r/w | 6B | VidActLeft(7:0) | | | | | | | |
| Active Video Width | r/w | 6F | VidActWidth(7:0) | | | | | | | |
| Active Video Height | r/w | 73 | VidActHeight(7:0) | | | | | | | |
| Video Win Top | r/w | 77 | VidWinTop(7:0) | | | | | | | |
| Video Win Left | r/w | 7B | VidWinLeft(7:0) | | | | | | | |
| Video Win Width | r/w | 7F | VidWinWidth(7:0) | | | | | | | |
| Video Win Height | r/w | 83 | VidWinHeight(7:0) | | | | | | | |

## 3.2 RGB Source Select.

The RGB source selection block determines which color plane is used for image position and level detection. The possible selections are RED, GREEN, BLUE, or the "OR" of all three. Table 3-2 illustrates the mapping of RGBSel to source selection.

Table 3-2 RGB Source Select Mapping

| RGBSel1 | RGBSel0 | Source |
|---|---|---|
| 0 | 0 | R or G or B |
| 0 | 1 | Red |
| 1 | 0 | Green |
| 1 | 1 | Blue |

## 3.3 Image Position Detect.

The image position block is responsible for detecting the left, right, top and bottom edges of the image. This information is used by the firmware to position the image properly on the LCD screen. Four eleven bit registers (top,bottom,left,right) get set each frame with the current values of the image extent for the selected color channel (see section 3.2). The top and bottom registers indicate the number of rows from VSync until the image starts and ends. The left and right registers indicate the number of columns from VidHFbk until the image starts and ends. An eight bit threshold register is set by firmware which determines how far from zero (black) a pixel must be before a start or end of image condition is determined. Some sources such as NTSC/PAL/SECAM will cause improper position detect since they have non-black data during blanking. Fortunately these are well defined standards where a constant image position setting can be selected. However, since there may be some installations where tricks of this kind are used there will be a software override for this logic.

## 3.4 Black & White Level Detect.

The black and white level detect module is used to determine the maximum and minimum values currently being digitized by the analog front end. Two types of black level are detected. The first is the lowest code on the DVB sampled durring blanking. The second is the actual black level of the image which is the lowest DVB code detected in the active area. Both these values are need so that firmware can determine how to adjust the reference. If the blanking level is slightly less than the black level the firmware assumes a small DC offset exists in the actual data and uses the black level value to adjust the ADC references. Otherwise, when these values differ by a large amount the firmware uses the blanking level. This is done because when there are large difference between these values then the image most likely contains no black data. To determine the white level the hardware monitors the data over the entire active area to determine the maximum pixel value. A comparator and holding register are used the implement this function. Once every VSync the eight bit software readable registers BlankLevel, BlackLevel and WhiteLevel are set. These registers may be read at anytime to obtain the current value of these levels.

## 3.5 Input Data Packer

The Input Data Packer converts the 24 bit wide pixels to 32 bit wide words by stuffing 4/3 pixels per word. Figure 3-2 illustrates the packing order for pixels into words. Note that the pattern repeats every three words. A basic fallout of the packing algorithm is that data is transferred to the input FIFO only three out of every four clock cycles.

| | | | | |
|---|---|---|---|---|
| Word 0 | R0 | G0 | B0 | R1 |
| Word 1 | G1 | B1 | R2 | G2 |
| Word 2 | B2 | R3 | G3 | B3 |

**Figure 3-2: Example of pixel packing.**

Another function of the Packer is horizontal image compression. This is done by selectively dropping pixels. The HDrop and HScale register specify whether this feature is enabled (HDrop = 1), and how often to pixel drop (HScale = 2 to 31, =2 means drop every other pixel, =31 means drop 1 out of 31 pixels). The corresponding expansion function is performed in the MOD block, while vertical compress/expand is done in the MAC block.

## 3.6 Input Fifo

The input data FIFO's main function is to absorb input data while transfer cycles are occurring thus preventing the loss of any data. This FIFO also bridges the asynchronous clock domains of the Input Video and WRAM. This allows the WRAM timing to remain consistent independent of the input video source. The depth of this FIFO is determined by the maximum length of time the output side can be stalled and the maximum input clock rate. Since the maximum sustainable input clock rate is related to input bandwidth of the WRAM, the FIFO depth can be derived as follows (refer to section 2.2):

$$\text{min FIFO depth} = (4/3)*((Sd-Rt*Ou)/Sd)*Rt \cong 19$$

Where: $Sd$ = SAM depth = 128 (16 bit words)

$Ou$ = Output bandwidth utilization = $(4/3) * 25/33 = 1$

$Rt$ = SAM refill time in number of clocks = 16

In the MID Block the FIFO depth is 48. Since the FIFO is after the Input Data Packer (see Section 3.5) the total width required is 32 bits.

### 3.6.1 Input Fifo Architecture

With a 43 MWord/Sec input data rate and a 50MWord/Sec output data rate the input data FIFO becomes a challenging design problem. To solve these design issues the three bank architecture illustrated in Figure 3-3 is used. The main advantage of this architecture is that the memories need only be single ported assuming that reads and writes are not allowed simultaneously to the same bank. This assumption simply implies that the write pointer is at least one bank ahead of the read pointer before reads can occur. This restriction requires that for each row of the display the number of words written must be divisible by 16 thus allowing the data at the end of each line to accessed. Since there are 4/3 pixels per word for a 640 pixel wide display a total of exactly 480 words would be written implying no extra writes would be required since 480 is evenly divisible by 16.

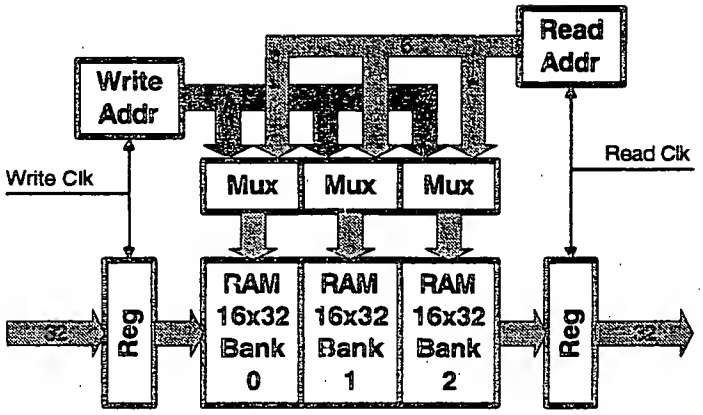



**Figure 3-3: Input Fifo block diagram.**

## 3.7 Word Compare Unit

The Word Compare Unit compares data read from the WRAM with the incoming packed pixel data. This comparison is performed on each byte of the 32 bit wide word. If any of the four bytes differs from the input data by more than the value loaded into the Threshold register then a counter is incremented. Once every VSync the counter is loaded into the MissCnt register and then reset. The circuit consists of four eight bit comparators and a fast (pixel clock rate) counter.

## 3.8 Image Capture Control.

The image capture controller frames the area in pixel space containing active pixel data. This unit uses the PLLClk and is not affected by the VidEn signal. In fact the output of this module VidActive is shorted to VidEn externally by the low cost computer interface card. The purpose of this unit is to specify to the hardware where valid data is located on the input side.

Figure 3-4 illustrates the various control registers used to control the position and size of the active data region. Note that the width and height can only be specified to within 8 pixels or lines.

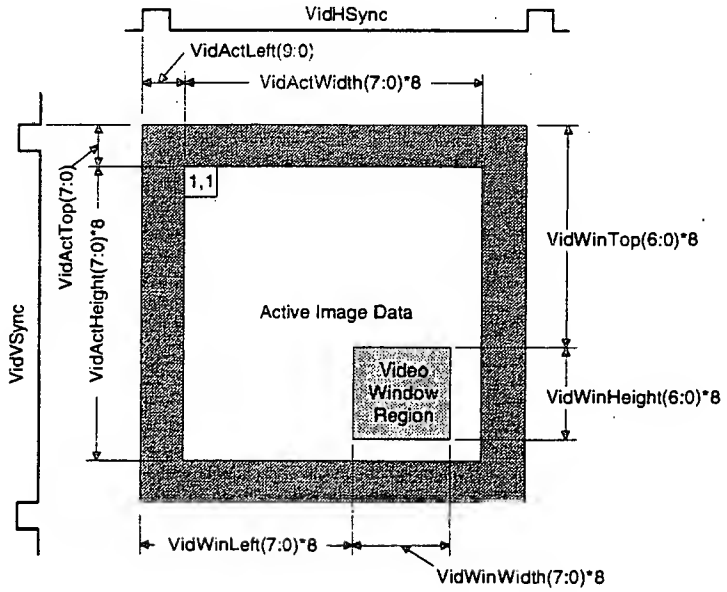



**Figure 3-4: Active Video Region Control.**

### 3.8.1 Video Window Control.

With an appropriately design video decoder Merlin is capable of supporting video windowing. The main requirement placed on the adapter is that it be capable of supplying data at the same rate as the DVB bus which could be as high as 65Mhz depending on the video mode. This is a rather extreme requirement and at the very least requires that the adapter has a frame store. A simple inversion of the DVB's source enable signal is used to select the video decoder when the input stream enters the video window region. The burden is placed on the video adapter to keep track of the current video stream's position by monitoring the DVB bus VidOE(n) and VidVSync signals.

### 3.8.2 Pixel Clock Control & Generation.

The three pixel clock sources are: Channel 1, Channel 2, and the PLL. The following figure illustrates the pixel clock distribution scheme used by Merlin. The PLL remains active while alternate clocks are selected as long as the correct sync source is selected in the MAC Block. This enables resizing modules to transfer data over the DVB at a fixed clock rate which is independent of the dot clock. The resizing engine can then use a FIFO and consume the blanking interval resulting in a slower clock rate into and out of the resizing chip.

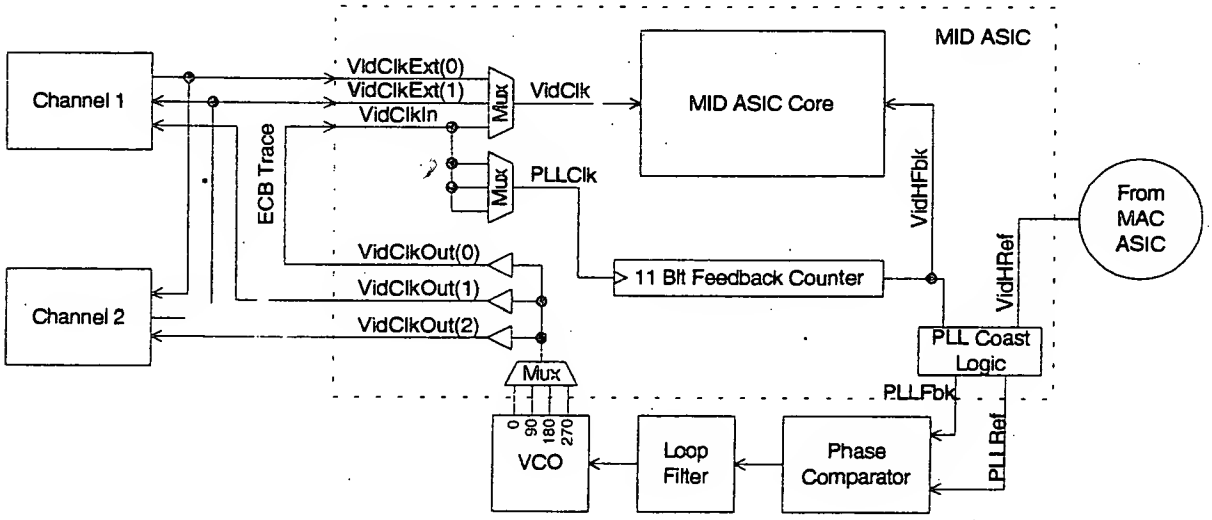



**Figure 3-5: Pixel Clock Generation.**

Note the careful clocking paradigm used by Merlin to ensure a sufficient setup and hold time for the external modules transfer of data to the Merlin ASIC. One important perfomance goal is to use digital PLL's to minimize skew between the on and off chip clocks. Also, the PLLClk and VidClk must be very closely aligned.

### 3.8.3 Issues with Composite Sync & PLL's.

A large number of composite sync signals have horizontal timing which varies significantly during the vertical interval. This unfortunately causes PLL's to unlock resulting in unstable system operation. In order to correct for this PLL updates are disabled during the vertical interval. The CoastEn bit is used to disable PLL updating during VSync. When CoastEn=1 the PLLFbk and PLLRef outputs are forced low starting when the current line number equals CoastStart, and ending CoastEnd lines after VSync thus preventing PLL updates.

### 3.8.4 Automatic Clock Phase Recovery (AutoSync).

Merlin supports three separate digital techniques for recovering the phase of the clock with respect to the data. Two of the techniques, frame compare & width detection, produce absolute indications of correct phase alignment, however, since they rely on specific aspects of the video data stream there are conditions where these techniques fail. When the firmware detects that these methods are not working a third technique, data edge detection, is used. Unfortunately there is a constant phase shift introduced by this technique which must be adjusted for each unit. Once set the phase shift will remain constant for all video sources which allows the unit to recover the clocks phase. The following three sections describe these techniques in greater detail with pseudo code. In addition these algorithms are only starting points there is great latitude for creating innovative firmware to solve this design problem. The goal of Merlin is to provide the necessary hardware resources to achieve the solution.

#### 3.8.4.1 Frame Compare AutoPhase.

Frame compare autosync is the classic LiteShow method which produces near perfect clock phase alignment and virtually guaranties the lowest visible noise in an image. Unfortunately this technique only works when the video data is not changing on a frame by frame basis. This approach will work if there are only small changes in the image such as moving the mouse, however, if a video window is active or the screen saver is running, or the computer is in the process of drawing to the screen this technique will fail. The good news is that the failure mode is simply an indication of no good clock phase (sync) settings. This technique can also fail with an all good clock phase indication when the screen contains no image data. As a result this technique should only be used when a video mode is first detected while the screen has been blanked and the hardware is in the process of self configuring. The pseudo code algorithm for frame compare autophase is detailed below.

```
#define  NumberOfSyncSettings 32
#define  MaxMissCount 2048

long int MissCountArray[NumberOfPhaseSettings];
int BestPhase;

/* Collect the sync data */
for(i=0;i<NumberOfPhaseSettings;i++) {
        SetPhaseDelay(i);
        WaitForVSync();
        Set CmprMode = 0;
        WaitForVSync();
        Set CmprMode = 1;
        WaitForVSync();
        MissCountArray[i] = Read of 20 bit Miss counter.
)
/* Find the best phase setting */
BestPhase = MinValue(MissCountArray);
if (!AllZero(MissCountArray) & (BestPase <= MaxMissCount)) {
        SetPhaseDelay(BestPhase);
} else {
        /* use a different method this one will not work. */
)
```

#### 3.8.4.2 Width Detection AutoPhase.

Width detection autophase compares the horizontal width of the input video source with the expected horizontal resolution stored in the mode table. If calculated value differs by two or less then a phase error is assumed and the phase delay is adjusted. This continues until the calculated and expected horizontal resolutions match or all phase delays have been tried. The draw back of this approach is that it depends on an image filling the screen horizontally which is virtually always the case when running Windows or Mac OS. However, if the user is running DOS then all bets are off and this is not the best approach to use. One significant advantage of this approach is that it can be used in an attempt to lock onto a miss-set tracking. For instance if the image is larger than the expected resolution the tracking is most likely wrong and the firmware can use the difference in expected vs. actual resolution to calculate a new tracking value. The pseudo code algorithm for frame compare autophase is detailed below. Note that this algorithm can be combined with the frame compare auto phase algorithm (shown above) into a

single loop which will cut in half the time to lock onto the correct phase. Also, the algorithm below only describes the auto-phase portion of the solution, auto-tracking is a relatively simple extension and is left to the reader.

```
#define  NumberOfSyncSettings 32

char  HDiffArray[NumberOfPhaseSettings];
int BestPhase;
int LeftEdge,RightEdge,HorzResolution;

HorzResolution = read mode table for current video modes horizontal resolution.
/* Collect the sync data */
for(i=0;i<NumberOfPhaseSettings;i++) {
        SetPhaseDelay(i);
        WaitForVSync();
        WaitForVSync();
        LeftEdge = read left edge image position register.
        RightEdge = read right edge image position register.
        HDiffArray[i] = RightEdge-LeftEdge-HorzResolution;
}
/* Find the best phase setting by finding the middle of the longest chain of zero vales in the HDiffArray. */
BestPhase = BestZero(HDiffArray);
if(BestPhase < 0) {
        /* No good phase setting found try changing the tracking. */
) else SetPhaseDelay(BestPhase);
```

### 3.8.4.3 Data Edge Detection AutoSync.

This circuit registers the data and a slightly delayed version of data every pixel clock. If the two values do not match then the a counter is incremented. The AutoEdge bit must be set to enable this circuit. The miss counter is updated every VSync with the new count value. The following block diagram illustrates the external circuitry required to implement the edge detect phase recovery method.

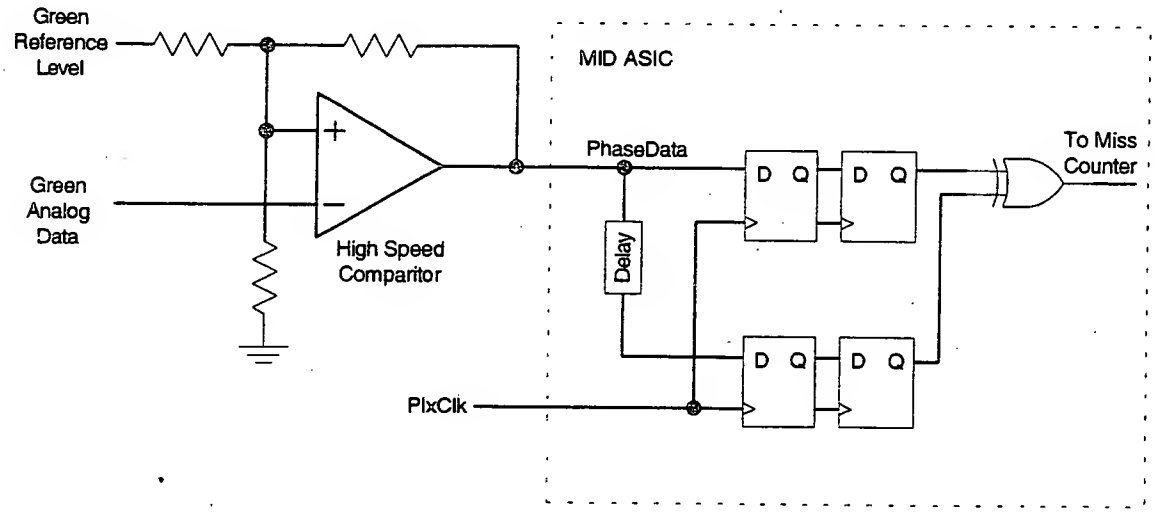



**Figure 3-6: Data Edge Detect AutoPhase Recovery Circuit.**

The main advantage of this method is that it does not depend on the image data like the other two methods do. However since there is an unknown phase shift caused by the comparator and the input signal path, the exact relationship between the clock and data cannot be determined. It would be relatively easy to provide a firmware based one time calibrate function that would determine the offset. When this circuit is enabled firmware would use the following algorithm to determine the optimal sync position.

```
#define  NumberOfSyncSettings 32
#define  MaxMissCount 2048

long int MIssCountArray[NumberOfPhaseSettings];
int BestPhase;

/* Enable edge detect auto phase recovery */
Set AutoEdge = 1;
/* Collect the sync data */
for(i=0;i<NumberOfPhaseSettings;i++) {
        SetPhaseDelay(i);
        WaitForVSync();
        WaitForVSync();
        MissCountArray[i] = Read of 24 bit Miss counter.
}
/* Find the best phase setting */
BestPhase = MinValue(MIssCountArray);
if (!AllZero(MissCountArray) & (BestPase <= MaxMissCount)) {
        SetPhaseDelay(BestPhase);
} else {
        /* use a different method this one will not work. */
}
```

## 3.9 MID Register Definitions

Table 3-3: MID Register Definitions

| Field | Offset | Bit(s) | r/w | Function |
|---|---|---|---|---|
| ChipRev(7:0) | 03 | 7:0 | r | Revision of the Merlin ASIC |
| PLLClkSel | 07 | 7 | r/w | Select the VidClk in pin as the video clock source. This is used for computer interface cards (except for External resizing such as Genisis). |
| IntFbkSel | 07 | 6 | r/w | Select the internal feedback counter for control of the PLL. |
| ClkEn(2:0) | 07 | 5:3 | r/w | Enable bits for each of the VidClk outputs.. |
| PLLPol | 07 | 2 | r/w | Determines the polarity of the PLL's horizontal reference. When set PLLHRef is active high. |
| RGBSel(1:0) | 07 | 1:0 | r/w | Selects which channel (R,G or B) is monitored for image position and black/white level detection. The channel selected is:<br>RGBSel1 RGBSel0 Channel<br>0 0 R or G or B<br>0 1 Red<br>1 0 Green<br>1 1 Blue |

| | | | | |
|---|---|---|---|---|
| WriteMask(7:0) | 0B | 7:0 | r/w | This register contains the value to be loaded into the WRAM write mask register. This is used to control on screen display. The hardware automatically inverts this register for CPU operations versus video capture cycles. For example when WriteMask = 0xfe the least significant bit of the R, G, and B fields is write protected for capture cycles. Since CPU cycles invert this register only the least significant bit would be written. |
| ThreshHold(7:0) | 0F | 7:0 | r/w | This register serves multiple functions. The image position logic uses the value in this register to indicate how far above black a pixel must be before it is counted as part of the image. The word compare circuit uses the value of this register to determine how different a pixels pervious value must be before it is considered to have changed. |
| BlkEdge | 13 | 7 | r/w | Selects which edge of BlackSample to use when detecting the blanking level. When zero the falling edge is used otherwise the rising edge is used. |
| AutoEdge | 13 | 6 | r/w | Enables the autosync edge detection algorithm. When this bit is set the MissCtr is incremented when ever the registered version of PhaseData and PhaseData delay do not agree. |
| VidWinEn | 13 | 5 | r/w | Video window enable. When set the channel data enable signals are inverted whenever the video stream enters the video window region. |
| HDrop | 17 | 7 | r/w | Enables horizontal pixel dropping. When set one out of every HScale valid pixels will not be written into the input FIFO. |
| VAvg | 17 | 6 | r/w | Determines the mode of operation of the input line dropper. When set averaging is performed by changing the line dropped on a frame by frame basis. |
| HAvg | 17 | 5 | r/w | Determines the mode of operation of the input pixel dropper. When set averaging is performed by changing the pixel dropped on a frame by frame basis. |
| HScale(4:0) | 17 | 4:0 | r/w | Determines the number of pixels over which the horizontal compress takes place (See HDrop definition). |

| | | | | |
|---|---|---|---|---|
| MissCnt(19:0) | 1B<br>1F<br>23 | 3:0<br>7:0<br>7:0 | r | This register indicates the number of pixels whose current frame value differs from the previous frames value by more than the amount indicated in the ThreshHold register. |
| Top(10:0) | 27<br>2B | 2:0<br>7:0 | r | This register is updated once per VSync and indicates the first line in the image which has data on the RGBSel channel which is greater than ThreshHold. |
| Bottom(10:0) | 2F<br>33 | 2:0<br>7:0 | r | This register is updated once per VSync and indicates the last line in the image which has data on the RGBSel channel which is greater than ThreshHold. |
| Left(10:0) | 37<br>3B | 2:0<br>7:0 | r | This register is updated once per VSync and indicates the first column in the image which has data on the RGBSel channel which is greater than ThreshHold. |
| Right(10:0) | 3F<br>43 | 2:0<br>7:0 | r | This register is updated once per VSync and indicates the last column in the image which has data on the RGBSel channel which is greater than ThreshHold. |
| BlankLevel(7:0) | 47 | 7:0 | r | This register is updated once per Vsync and indicates the black level durring blanking for the RGBSel channel. |
| BlackLevel(7:0) | 4B | 7:0 | r | This register is updated once per VSync and indicates the black level durring the active image area for the RGBSel channel. |
| WhiteLevel(7:0) | 4F | 7:0 | r | This register is updated once per VSync and indicates the white level for the RGBSel channel. |
| ClksPerLine(10:0) | 53<br>57 | 2:0<br>7:0 | r/w<br>r/w | This register contains the number of pixel clocks in a video line. It is used to load the divide by N counter of the PLL once every horizontal reference. |
| CoastEn | 5B | 7 | r/w | Enables PLL coasting. When set PLL updates will be turned off starting at line CoastStart(10:0) and turned back on starting CoastStop(3:0) Lines after VidVSync. |
| CoastStop(3:0) | 5B | 6:3 | r/w | Number of lines after trailing edge of VidVSync until PLL coasting is turned off. |
| CoastStart(10:0) | 5B<br>5F | 2:0<br>7:0 | r/w<br>r/w | Number of line following trailing edge of VidVSync until the PLL is turned off. |

| | | | | |
|---|---|---|---|---|
| VidActTop(7:0) | 5B | 7:0 | r/w | Set the vertical position of the image. This field contains the number of lines from the leading edge of VidVSync until the active video starts (i.e., vertical blanking ends). |
| VidActLeft(9:0) | 5F<br>63 | 1:0<br>7:0 | r/w | Set the horizontal position of the image. This field contains the number of lines from the leading edge of VidHFbk until the active video starts (i.e., horizontal blanking ends). |
| VidActWidth(7:0) | 67 | 7:0 | r/w | This register specifies the number of clocks divided(i.e., pixels/8) of active data in a video line. |
| VidActHeight(7:0) | 6B | 7:0 | r/w | This register specifies the number of lines divided(i.e., pixels/8) of active data in a video frame. |
| VidWinTop(7:0) | 6F | 7:0 | r/w | This register specifies the number of lines divided by eight from the leading edge of VSync until the video in a window region starts. |
| VidWinLeft(7:0) | 73 | 7:0 | r/w | This register specifies the number of clocks divided by eight from the leading edge of VidHFbk until the video in a window region starts. |
| VidWinWidth(7:0) | 77 | 7:0 | r/w | This register specifies the width of the video in a window region divided by eight. |
| VidWinHeight(7:0) | 7B | 7:0 | r/w | This register specifies the height of the video in a window region divided by eight. |

# 4. MERLIN OUTPUT DATAPATH (MOD)

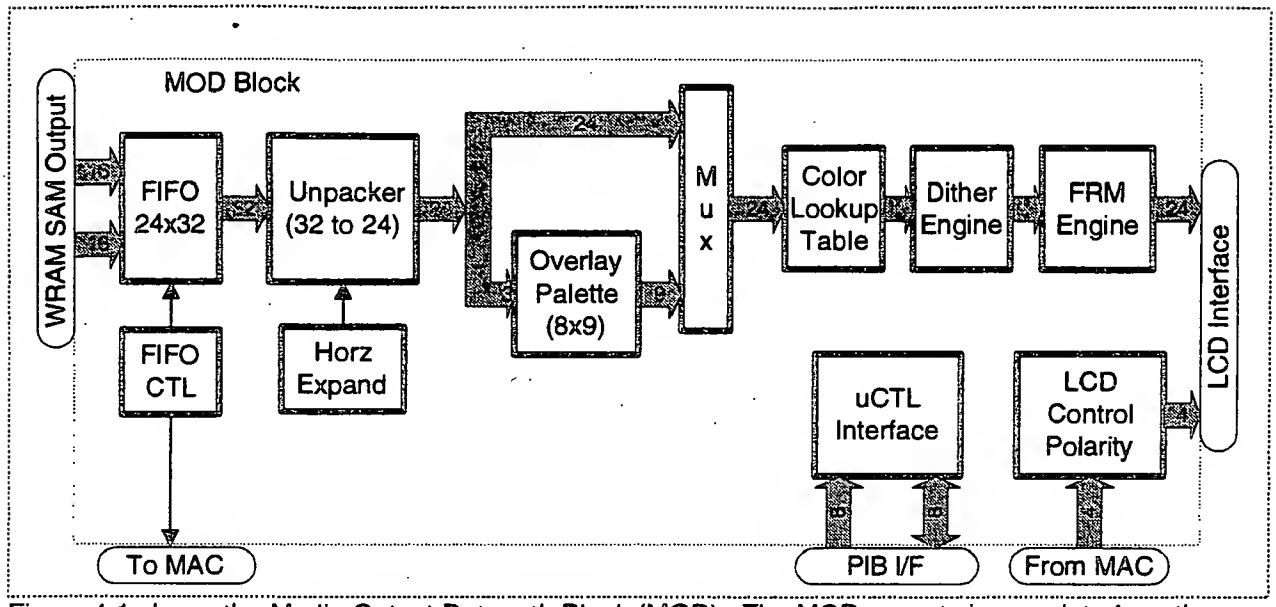



Figure 4-1 shows the Merlin Output Datapath Block (MOD). The MOD accepts image data from the WRAM frame buffer, and formats it properly for display on the LCD. This chip contains the following functions. frame buffer output FIFO and pixel unpacker, menu overlay palette and control, color lookup table (gamma correction), spatial dithering, and frame-rate-modulation (grey scaling). These last functions perform digital signal processing on the image data to improve color rendering and increase the percieved number of colors displayed on the LCD to nearly 16 million regardless of the LCD type.

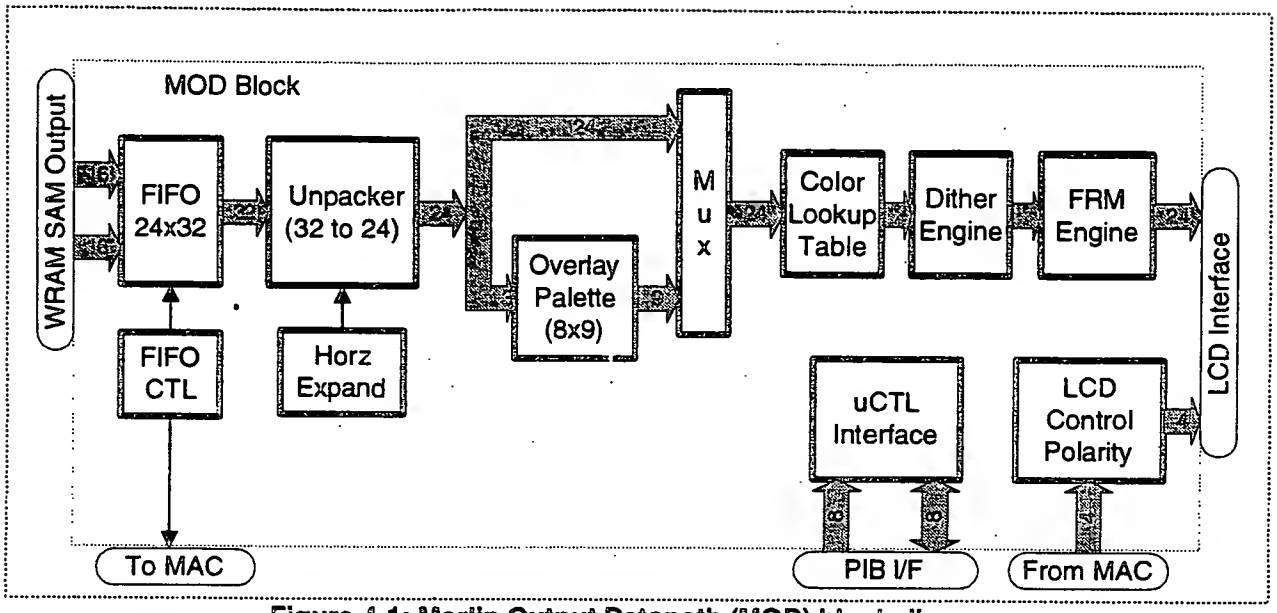



**Figure 4-1: Merlin Output Datapath (MOD) block diagram.**

The MOD supports single-scan (not dual-scan) VGA (640X480) and SVGA (800x600) LCD's with 3, 4, 6, or 8-bit drivers and single-pixel wide data paths. For SVGA LCD's, two 16-bit data paths (from even and odd WRAM's) are combined into a single data path. For VGA LCD's, only one 16-bit data path (only one WRAM) is used.

## 4.1 MOD Register Address Map.

The programmable registers contained in the MOD are shown in Table 4-1. The MOD occupies a 4Kbyte address space. In this 4K space, only every 4th byte is used to keep the data aligned to the rightmost byte in a 32 bit word (ie 3,7,B,F, ...). This results in 1K register locations. The first 256 locations are where the general registers reside, and the last 768 locations are occupied by the Color Lookup Table. The base address of the MOD is determined by external address decoding logic via the chip select input.

Table 4-1 MOD Register Map

| Base+3E0000+Offset | | | Bit Position | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Function | r/w | Offset | 7 | 4 | 5 | 4 | 3 | 2 | 1 | 5 |
| LCD Ctl | r/w | 003 | | | | CLUTBy | LCDVSPol | LCDHSPol | LCDDEPol | LCDClkPol |
| FRM Control | r/w | 007 | | | | FrmDiv | FrmMode(1:0) | | FrmPos(1:0) | |
| Dither Control | r/w | 00B | | | | | DitMode(1:0) | | DitPos(1:0) | |
| Horz Expand | r/w | 00F | HExpEn | | | HExpScale(4:0) | | | | |
| FRM Data | r/w | 103-1FF | | | | | | | FRMData(1:0) | |
| Overlay Palette | r/w | 203-25F | | | | | | OvlData (2:0) | | |
| Red Color Lookup Table | r/w | 403-7FF | RedCLUT (7:0) | | | | | | | |
| Green Color Lookup Table | r/w | 803-BFF | GrnCLUT (7:0) | | | | | | | |
| Blue Color Lookup Table | r/w | C03-FFF | BluCLUT (7:0) | | | | | | | |

## 4.2 Output FIFO

The Output FIFO is shown in Figure 4-2. The Output FIFO's input is connected to the WRAM serial access memory port (SAM), the output of the FIFO feeds the rest of the MOD chip's data path which provides data to the LCD. The purpose of this FIFO is decouple the output of the WRAM (which runs at a fixed frequency) from the LCD output clock rate which may not be as fast.

The FIFO has two modes of operation - VGA LCD mode and SVGA LCD mode. In VGA LCD mode, there is only one WRAM, and the SAM data from the WRAM is demultiplexed into a 32 bit wide data stream before writing into the FIFO. In SVGA LCD mode, there are two WRAM's, and the data from both SAM's is written into the FIFO simultaneously.

The FIFO is divided into three 8 x 32 bit banks, each of which is single ported (can be either read or written at any give time - not both at once).

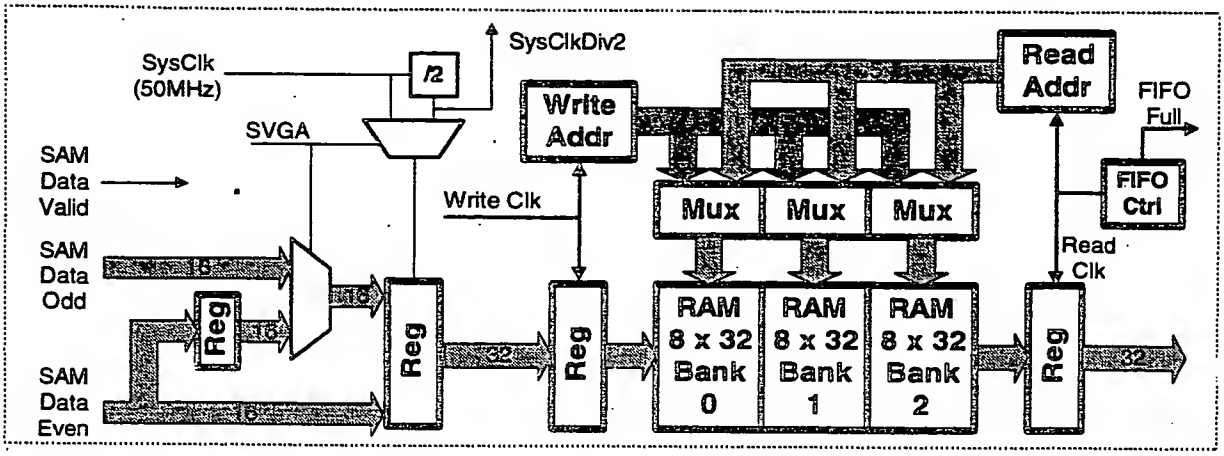



**Figure 4-2 Output FIFO**

### 4.2.1 FIFO Bandwidth

In VGA mode, the WRAM output runs at a maximum bandwidth of 2 bytes every 20nS (50MHz) = 100Mbytes/sec. This is equivalent to 33.3Mpixels/sec (3 bytes/pixel). This rate is guaranteed to be sustainable because the WRAM control logic pauses the WRAM input port whenever the WRAM SAM port needs more data. The FIFO output cannot exceed the input rate, and typically needs to run at 25 to 30Mpixels/sec to meet the timing requirements of most VGA TFT LCD displays. This bandwidth is only required during active display lines, and pauses for every line (Horz. blanking) and every frame (Vertical blanking). To reduce system cost, the LCDClk can be derived from 1/2 of the SysClk rate - 25MHz. For a typical VGA TFT LCD, this results in a maximum vertical refresh rate of about 63Hz.

In SVGA mode, the FIFO input bandwidth is doubled to 66.6Mpixels/sec since two WRAM SAM's can provide 32 bits of data every 20nS (50mhz * 4/3 pixels = 66.6). The FIFO output cannot exceed this rate, and typically runs at 40Mpixels/sec to work with most SVGA TFT LCD's.

### 4.2.2 FIFO Input Section

In VGA mode, the initial stage of the FIFO demultiplexes the 16 bit data stream from the WRAM SAM to a 32 bit word running at 25MHz. Next the data is stored into one of the FIFO banks. Data is always written into the FIFO in complete banks. At the end of a bank write, the FIFO control logic tells the MAC chip whether another bank is available for writing or if it is necessary to pause until a bank becomes available for writing. This is done by means of the OutFIFOFull signal. The arrangement of the pixel data in the FIFO is shown in Table 4-2. The number indicates the pixel number (starting at 0), the subscript refers to the RGB bytes that make up each pixel.

| FIFO Bank Word | Byte 3 | Byte 2 | Byte 1 | Byte 0 |
|---|---|---|---|---|
| 0 | $0_{RED}$ | $0_{GRN}$ | $0_{BLU}$ | $1_{RED}$ |
| 1 | $1_{GRN}$ | $1_{BLU}$ | $2_{RED}$ | $2_{GRN}$ |
| 2 | $2_{BLU}$ | $3_{RED}$ | $3_{GRN}$ | $3_{BLU}$ |
| 3 | $4_{RED}$ | $4_{GRN}$ | $4_{BLU}$ | $5_{RED}$ |
| 4 | $5_{GRN}$ | $5_{BLU}$ | $6_{RED}$ | $6_{GRN}$ |
| 5 | $6_{BLU}$ | $7_{RED}$ | $7_{GRN}$ | $7_{BLU}$ |
| 6 | $8_{RED}$ | $8_{GRN}$ | $8_{BLU}$ | $9_{RED}$ |
| 7 | $9_{GRN}$ | $9_{BLU}$ | $10_{RED}$ | $10_{GRN}$ |

In SVGA mode, both of the 16 bit WRAM SAM data streams are written into a FIFO bank at 50MHz. The arrangement of the pixel data in the FIFO for this mode is the same as VGA mode (shown above) except for the fact that the FIFO fills up twice as quickly.

The FIFO input section is reset at the beginning of each LCDHsync period. On the leading edge of LCDHsync all banks of the FIFO are marked as empty so they can be filled in preparation for the next line.

When data is sent to the FIFO, it is accompanied by SAMDataValid. This informs the FIFO that the pixels it is currently receiveing should be stored. This signal will be active for 16 consecutive clocks in VGA mode, or 8 clocks in SVGA mode (corresponds to loading one FIFO bank).

### 4.2.3 FIFO Output Section

The FIFO output section runs off of the LCDClk. Because the FIFO is 32 bits wide, 3 FIFO output words (32 bits wide) contain 4 pixels (24 bits/pixel) worth of data. This means that the FIFO ouput pauses for one cycle out of every 4. This is shown below in Figure 4-3.

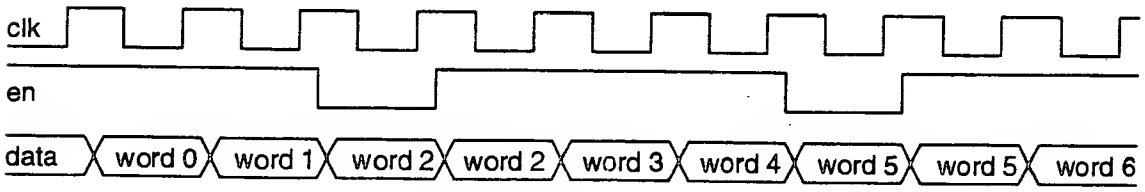



**Figure 4-3: FIFO Output Data Flow**

During normal operation (not rear project), each bank is read from word 0 to word 7. In the case of horizontal reverse (rear project), each FIFO bank is read out from word 7 to word 0. The data read from the FIFO goes to the unpacker to reassembled into pixels.

### 4.2.4 FIFO Handshaking

Because of the bank nature of the FIFO, once a bank has been read, the entire bank is available for writing. The FIFO control logic keeps the OutFIFOFull flag set (= 1) until a bank becomes available for writing. At this time, the flag is deasserted (=0), which tells the MAC to start issuing SAM data to the FIFO input. Depending on whether the MOD is in VGA or SVGA mode, the MOD will keep the OutFIFOFull flag low for 16 or 8 SysClk cycles. This will fill an entire FIFO bank with new SAM data (8 x 32 bits). If the next bank is also available for writing, the OutFIFOFull flag will stay deasserted (in 16 or 8 SysClk increments) until all the available FIFO banks are written.

When valid data is sent to the FIFO, SAMDataValid will be active to indicate this condition.

## 4.3 Unpacker

The Unpacker takes the 32 bit words from the FIFO and reassembles this data into a continuous stream of pixels. The Unpacker block diagram is shown in Figure 4-4.

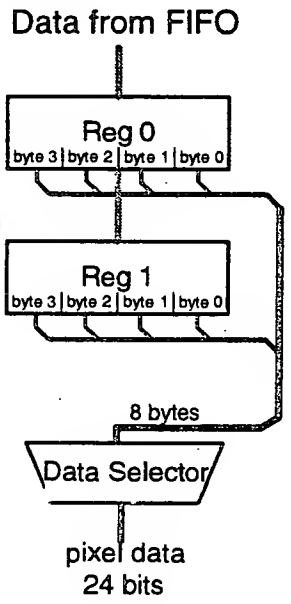



**Figure 4-4: Unpacker Block Diagram**

The Unpacker consists of a two stage pipeline, with each stage being 32 bits wide. Following the pipeline is the data selector mux which routes the bytes to the proper position. This mux is capable of taking any of the 8 available input bytes and routing it to one of the 3 output bytes. The determination of the byte unpacking order depends on the following conditions: byte position in the current video line, and horizontal flip mode (rear project).

### 4.3.1 Horzontal Expansion

Horizontal image expansion is performed in the Unpacker by selectively replicating pixels. This function is used for crude image resizing to allow smaller images to fill the entire LCD screen. Horizontal expansion is controlled by the HExpScale count. The value of this register determines how often to replicate a pixel. The valid values are 1 to 31. If the HExpScale =1, then every pixel is replicated, if the HExpScale=31, then one pixel out of 31 is replicated. The corresponding Horizontal image compression function is done in the MID block by selectively not storing pixels in the WRAM. Vertical expansion/compression is controlled by the MAC, by selectively replicating or skipping lines in the WRAM.

## 4.4 Overlay Palette

The Overlay Palette contains the color values for the on screen display used for menus and scribble mode. The organization of the Overlay Palette is show in Table 4-3: Overlay Palette.

Table 4-4: Overlay Palette

| Palette Word | Red | Green | Blue | Typical Values | Resultant Color |
|---|---|---|---|---|---|
| 1 | RRR00000 | GGG00000 | BBB00000 | 00 00 00 | Black |
| 2 | RRR00000 | GGG00000 | BBB00000 | E0 00 00 | Red |
| 3 | RRR00000 | GGG00000 | BBB00000 | 00 E0 00 | Green |
| 4 | RRR00000 | GGG00000 | BBB00000 | 00 00 E0 | Blue |
| 5 | RRR00000 | GGG00000 | BBB00000 | 60 60 60 | Dark Grey |
| 6 | RRR00000 | GGG00000 | BBB00000 | A0 A0 A0 | Light Grey |
| 7 (virtual) | . 11111111 | 11111111 | 11111111 | FF FF FF | White |

The Overlay Palette memory consists of 6 entries, each having 9 bits. The 9 bits in each entry are used as the three most significant bits of red, green, and blue when the overlay is selected to be displayed. These values are loaded by the software as described in the register definition (Table 4-1). Typical values for the Overlay Palette are shown in Table 4-4. The seventh entry in the Overlay Palette is a special case - it is hardcoded to full white and is not user programmable.

## 4.5 On Screen Display Multiplexer

The On Screen Display Multiplexer (OSD Mux) is responsible for selecting whether to display the normal pixel data or the overlay data, on a pixel by pixel basis. The OSD Mux is only used when overlay mode is enabled (OverlayEn = 1). The definition of the selection bits used by the OSD Mux is shown in Table 4-4.

Table 4-4: On Screen Display Multiplexer Selection Bits

| Mode | Red | | | | | | | | Green | | | | | | | | Blue | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OvlEn = 0 | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 | G7 | G6 | G5 | G4 | G3 | G2 | G1 | G0 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| OvlEn = 1 | R7 | R6 | R5 | R4 | R3 | R2 | R1 | O2 | G7 | G6 | G5 | G4 | G3 | G2 | G1 | O1 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | O0 |

When OverlayEn = 0, all 8 bits of data for each pixel is sent on to the CLUT (color lookup table). In this case it is possible to display 16.7M colors with some LCD types. When OverlayEn = 1, the LSB of each color plane is defined to be one of the OSD Mux selection bits. In this case there are only 7 bits of image data per color plane. Generally this does not reduce the color gamut of the image, because the image has already been dithered in the MID (reduced from 8 bits to 7, 6, or 5 bits). In general practice, overlay mode can be left on all the time which speeds up menu writes slightly. The OSD Mux selection function is defined in Table 4-5.

Table 4-5: OSD Mux Selection Function

| OSD Select Bits (O2 - O0) | Function | Data sent to CLUT |
|---|---|---|
| 000 | transparent | 7 MSBs of RGB |
| 001 - 111 | overlay | Overlay Palette |

If the select bits for a pixel are all 0, then the MS 7 bits of RGB are allowed to pass through the mux unchanged. If the select bits are non-zero, then the Palette Overlay memory is selected instead (the MS 7 bits of RGB are ignored in this case). The entry in the Palette Overlay to use is determined by the select bits (001 to 111).

By writing the appropriate data into the WRAM frame buffer and enabling the overlay mode, the software can control whether any given pixel is displayed or overlayed. This is useful for drawing the on screen menus, scribble mode, and on screen pointers.

## 4.6 Color Lookup Table

The Color Lookup Table (CLUT) provides a very flexible mechanism for affecting the video data in a number of ways. These functions include:

- LCD gamma correction
- decreasing the color palette
- brightness & contrast adjustment
- "fade to black" dissolves for Liteshow
- "shifting color" special effects on startup screens

The CLUT block diagram is shown in Table 4-5 below.

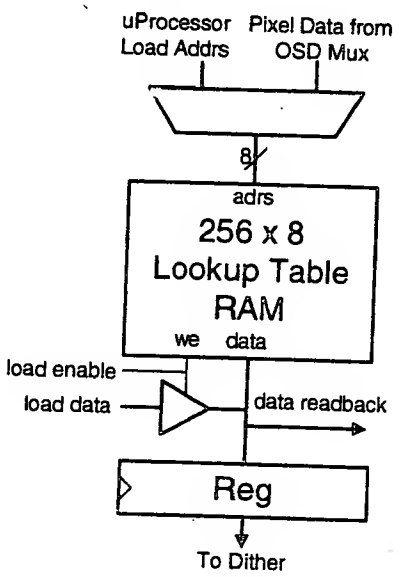



**Figure 4-5: Color LookupTable (one color plane shown)**

The CLUT consists of 3 256 x 8 bit RAM's - one for each color plane. The pixel data is used as the address into the CLUT and the resultant data output is routed to the dithering and FRM blocks. The CLUT RAM is loaded by the microprocessor according to the format in the register description (Table 4-14). During loading, the address and data are supplied by the microprocessor. To avoid screen artifacts this must be done during LCD blanking.

## 4.7 Spatial Dither Engine

The dither engine provides the illusion of more levels of intensity for each color plane by rounding a pixel's intensity up or down to a representable level based on it's x-y coordinates. The MOD supports 1,2, or 3 bit dithering of the input pixel data. The term "1 bit" means that the pixel data is reduced in width by 1 bit (i.e. 8 bit pixel data, becomes 7 bits wide after 1 bit dithering). A summary of the dithering modes available is shown in the following table..

Table 4-5: DitherMode

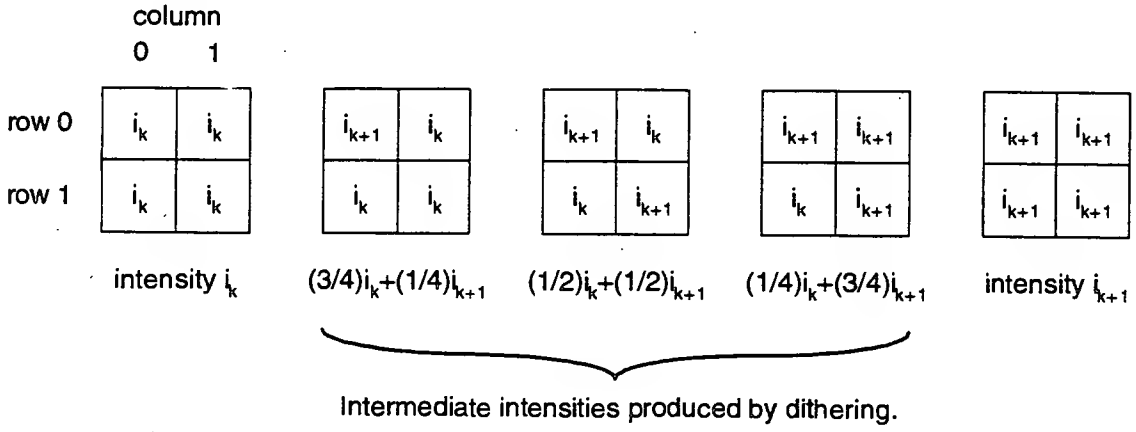
| Dithering Mode | DitMode | # of intermediate shades | Size of dither grid used |
|---|---|---|---|
| off | 00 | 0 | none |
| 1 bit | 01 | 1 | 2 x 2 (subset of 2 bit) |
| 2 bit | 10 | 3 | 2 x 2 |
| 3 bit | 11 | 7 | 3 x 3 |

The number of intermediate shades generated refers to the number of shades created between each adjacent FRM (frame rate modulation) shade. The dither grid size describes the x-y size of the spatial grid used to generate the dither patterns. For instance, on a 3x3 grid, the dither pattern repeats every 3 rows and 3 columns for a given input pixel data value. The dithering mode is set by programming the DitherMode bits.

### 4.7.1 Dither Algorithm

The following examples details the operation of 2 & 3 bit dithering to acheive intermediate shades.

In the 2x2 case if there are n-levels of intensity which can be achieved via a combination of FRM (frame rate modulation) and direct drive then the total number of levels of intensity for large areas can be increased to 4n-3. Figure 4-6 and Figure 4-7 illustrates the algorithm used to create intermediate intensities in an arbitrary four & nine pixel square area. This technique relies on the eye to integrate the area into a average intensity. Generally speaking, the smaller the difference in the dither intensity levels, the better the overall result. It should also be note that this technique is purely spatial and does not rely on propagation of error terms (i.e., error diffusion). An error diffusion algorithm would produce superior results at a cost of significant hardware resources.




Intermediate intensities produced by dithering.

**Figure 4-6: 2x2 Dither Algorithm Overview.**

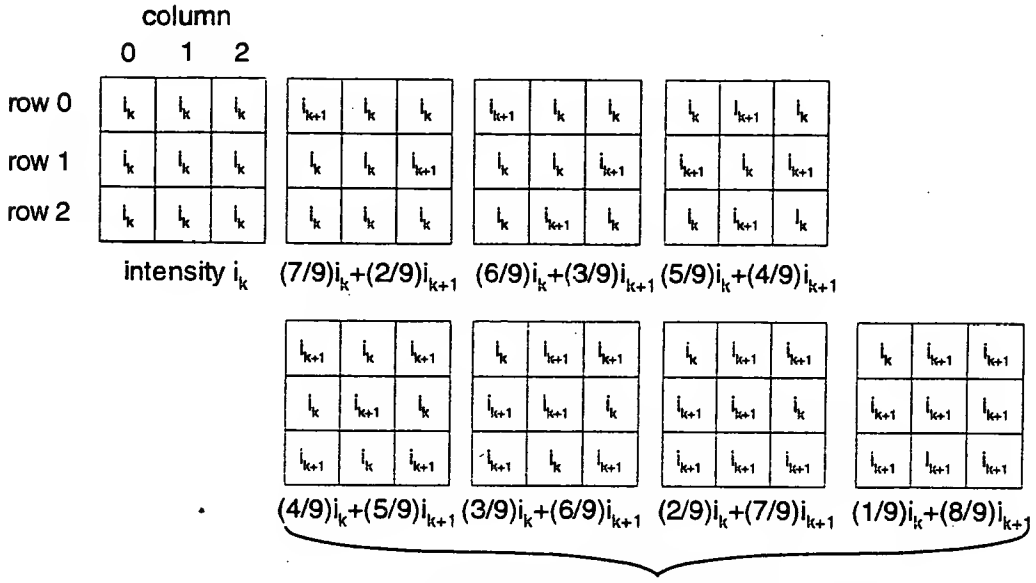



**Figure 4-7: 3x3 Dither Algorithm Overview.**

The bit position of the color plane which is used to select the dithering pattern is programmable allowing for various color resolutions of the display device. The DitPos bits in conjuction with the DitherMode used to control this selection (see Table 4-7).

**Table 4-7: Dither source bits selection.**

| DitMode(1:0) | DitPos(1:0) | Bit's Selected |
|---|---|---|
| 00 | xx | N/A |
| 01 | 00 | 0 |
| 01 | 01 | 1 |
| 01 | 10 | 2 |
| 01 | 11 | 3 |
| 10 | 00 | 1,0 |
| 10 | 01 | 2,1 |
| 10 | 10 | 3,2 |
| 10 | 11 | 4,3 |
| 11 | 00 | 2,1,0 |
| 11 | 01 | 3,2,1 |
| 11 | 10 | 4,3,2 |
| 11 | 11 | undefined |

For example, with a four bit LCD where FRM is used to extend the number of intensity levels to 6 bits then selecting 2 bit dithering (DitherMode = 10) results in a total of eight bits of intensity resolution. And the DitPos would be set to '00' to select the correct pixels. Since there are three color planes this would result in approximately 83 or 16.7million (A more accurate number would be (4*(4*16-3)-3)3 which equals ~14million). The following tables (Table 4-8 & Table 4-9) indicate how the selected pixel bits are used to choose a dither pattern. Note that this mapping produces an end case where the all one's code would end up dithered preventing solid fields of the max intensity level. However, since there are three

less intensities than the total binary space (this results from a 4n-3 mapping produced by FRM algorithms) the dither pattern simply selects the same intensity level at the extremes resulting in a pure white.

Table 4-8: 2x2 Dither Mapping

| Sel1 | Sel0 | 1X1 Function | 2X2 Function |
|---|---|---|---|
| 0 | 0 | $i_k$ | $i_k$ |
| 0 | 1 | $i_k$ | $(3/4)i_k+(1/4)i_{k+1}$ |
| 1 | 0 | $(1/2)i_k+(1/2)i_{k+1}$ | $(1/2)i_k+(1/2)i_{k+1}$ |
| 1 | 1 | $(1/2)i_k+(1/2)i_{k+1}$ | $(1/4)i_k+(3/4)i_{k+1}$ |

Table 4-9: 3x3 Dither Mapping

| Sel2 | Sel1 | Sel0 | Function |
|---|---|---|---|
| 0 | 0 | 0 | $i_k$ |
| 0 | 0 | 1 | $(7/9)i_k+(2/9)i_{k+1}$ |
| 0 | 1 | 0 | $(6/9)i_k+(3/9)i_{k+1}$ |
| 0 | 1 | 1 | $(5/9)i_k+(4/9)i_{k+1}$ |
| 1 | 0 | 0 | $(4/9)i_k+(5/9)i_{k+1}$ |
| 1 | 0 | 1 | $(3/9)i_k+(6/9)i_{k+1}$ |
| 1 | 1 | 0 | $(2/9)i_k+(7/9)i_{k+1}$ |
| 1 | 1 | 1 | $(1/9)i_k+(8/9)i_{k+1}$ |

## 4.8 Frame Rate Modulation Engine

The Frame Rate Modulation Engine (FRM) increases the perceived number of displayable colors by varying the displayed data on a frame by frame basis. FRM is very similar to dithering, except where dithering works over the x-y space within a single frame, FRM rounds the data value up and down from one frame to the next. Because the LCD refresh rate is high (generally 60Hz or faster), the observer's eye averages the FRM intensity shifts from one frame to the next into "new" intermediate levels of brightness.

The FRM has four modes of operation - 1 bit FRM, 2 bit FRM and 1/2 speed mode for each of the previous modes. Using 1 bit FRM, the FRM engine operates over 2 successive frames and produces one intermediate shade. Using 2 bit FRM, the FRM engine operates over 4 successive frames and

produces three intermediate shades. The 1/2 speed mode (FrmDiv = 1), causes the FRM engine to repeat each FRM pattern for two successive frames. In this case the 1 bit FRM operates over 4 frames (2 frames in one state, then two frames in the other state) and the 2 bit FRM operates over 8 frames (2 frames in each of the 4 states). This mode will increase the amount of flicker slightly, but it prevents the accumulation of DC bias on some TFT LCD's (known as "patterning" phenomenon).

### 4.8.1 FRM Shading Patterns

The FRM also uses some spatial dispersion to reduce "shading noise" artifacts. This is done with a 4x4 grid similar to the dithering grids described above. To optimize the performance with any LCD, the FRM shading grids are programmable as described in Table 4-14. Some example FRM shade patterns are shown in Table 4-10 and Table 4-11. Note the FPM algorithm is programmable down to the RGB subpixel (I.e., the Red, Green, and Blue planes can use different algorithms). This is necessary because some displays require an offset in the algorithm between the color subpixels.

**Table 4-10: Half Density FRM Shade Pattern**

frame 0

| | col0 | col1 | col2 | col3 |
|---|---|---|---|---|
| row0 | | 1 | | 1 |
| row1 | | 1 | | 1 |
| row2 | 1 | | 1 | |
| row3 | 1 | | 1 | |

frame 1

| col0 | col1 | col2 | col3 |
|---|---|---|---|
| 1 | | 1 | |
| 1 | | 1 | |
| | 1 | | 1 |
| | 1 | | 1 |

**Table 4-11: Quarter Density FRM Shade Pattern**

frame 0

| | col0 | col1 | col2 | col3 |
|---|---|---|---|---|
| row0 | 1 | | 1 | |
| row1 | | | | |
| row2 | | 1 | | 1 |
| row3 | | | | |

frame 1

| col0 | col1 | col2 | col3 |
|---|---|---|---|
| | | | |
| 1 | | 1 | |
| | | | |
| | 1 | | 1 |

frame 2

| col0 | col1 | col2 | col3 |
|---|---|---|---|
| | | | |
| | 1 | | 1 |
| | | | |
| 1 | | 1 | |

frame 3

| col0 | col1 | col2 | col3 |
|---|---|---|---|
| | 1 | | 1 |
| | | | |
| 1 | | 1 | |
| | | | |

In 1 bit FRM mode, only the half shade pattern is used. In the 2 bit FRM mode, both the half shade and the quarter shade patterns are used. The 3/4 intensity shade is just the inverse of the quarter shade pattern.

### 4.8.2 FRM Example: 3 bit LCD with 1 bit FRM

This example illustrates how intermediate shades are generated using the case of a 3 bit LCD (capable of displaying $8^3 = 512$ native colors) and 1 bit FRM to produce 3375 displayable colors. The 1 bit FRM produces 1 intermediate shade inbetween each of the 8 native shades for a total of 15 shades per color plane. This results in a total of 15*15*15 = 3375 colors as shown in Table 4-12 below.

**Table 4-12: 1 bit FRM Example**

| Native LCD shades | 1 bit FRM generated shades | New shade numbers |
|---|---|---|
| 7 | | 14 |
| | 6*.5 + 7*.5 | 13 |
| 6 | | 12 |
| | 5*.5 + 6*.5 | 11 |
| 5 | | 10 |
| | 4*.5 + 5*.5 | 9 |
| 4 | | 8 |
| | 3*.5 + 4*.5 | 7 |
| 3 | | 6 |
| | 2*.5 + 3*.5 | 5 |
| 2 | | 4 |
| | 1*.5 + 2*.5 | 3 |
| 1 | | 2 |
| | 0*.5 + 1*.5 | 1 |
| 0 | | 0 |

### 4.8.3 Allowable Combinations of Dithering and FRM

Table 4-13 illustrates the allowable combinations of dithering and FRM. Combinations are shown for 3 bit, 4 bit and 6 bit LCD's. "L" represents bits that go to the LCD. "F" indicates bits used for FRM. "D" is for bits used for dithering. All these are additive, so that if "D", "F", and "L" are used together this means that the "D" bits are used for dithering which affects all more significant "F" and "L" bits. In tum, the "F" bits are used for FRM which affects all more significant "L" bits.

## Supported Dithering/FRM Combinations

| # Bits Used | # Shades Generated | Perceived Colors | Bit mapping before CLUT (7 6 5 4 3 2 1 0) | Dither Pos | Dither Mode | FRM Mode | FRM Pos |
|---|---|---|---|---|---|---|---|
| **3 bit LCD** | | | | | | | |
| 3 | $8^3$ | 512 | L L L | X | 00 | X | 00 |
| 4 | $(2*8-1)^3 = 15^3$ | 3,375 | L L L F | X | 00 | 00 | 11 |
| 5 | $(4*8-3)^3 = 29^3$ | 24,389 | L L L F D | 11 | 01 | 00 | 11 |
| 6 | $(4*8-3)^3 = 29^3$ | 24,389 | L L L F F | X | 00 | 01 | 11 |
| 7 | $(4*8-3)^3 = 29^3$ | 24,389 | L L L D D | 11 | 10 | X | 00 |
| 6 | $(8*8-7)^3 = 57^3$ | 185,193 | L L L F D D | 10 | 10 | 00 | 11 |
| 6 | $(8*8-7)^3 = 57^3$ | 185,193 | L L L F F D | 10 | 01 | 01 | 11 |
| 6 | $(8*8-7)^3 = 57^3$ | 185,193 | L L L D D D | 10 | 11 | X | 00 |
| 7 | $(16*8-15)^3 = 113^3$ | 1,442,897 | L L L F D D D | 01 | 11 | 00 | 11 |
| 7 | $(16*8-15)^3 = 113^3$ | 1,442,897 | L L L F F D D | 01 | 10 | 01 | 11 |
| 8 | $(32*8-31)^3 = 225^3$ | 11,390,635 | L L L F F D D D | 00 | 11 | 01 | 11 |
| **4 bit LCD** | | | | | | | |
| 4 | $16^3$ | 4096 | L L L L | X | 00 | X | 00 |
| 5 | $(2*16-1)^3 = 31^3$ | 29,791 | L L L L F | X | 00 | 00 | 10 |
| 5 | $(2*16-1)^3 = 31^3$ | 29,791 | L L L L D | 11 | 01 | X | X |
| 6 | $(4*16-3)^3 = 61^3$ | 226,981 | L L L L F D | 10 | 01 | 00 | 10 |
| 6 | $(4*16-3)^3 = 61^3$ | 226,981 | L L L L F F | X | 00 | 01 | 10 |
| 6 | $(4*16-3)^3 = 61^3$ | 226,981 | L L L L D D | 10 | 10 | X | X |
| 7 | $(8*16-7)^3 = 121^3$ | 1,771,561 | L L L L F D D | 01 | 10 | 00 | 10 |
| 7 | $(8*16-7)^3 = 121^3$ | 1,771,561 | L L L L F F D | 01 | 01 | 01 | 10 |
| 7 | $(8*16-7)^3 = 121^3$ | 1,771,561 | L L L L D D D | 01 | 11 | X | X |
| 8 | $(16*16-15)^3 = 241^3$ | 13,997,521 | L L L L F D D D | 00 | 11 | 00 | 10 |
| 8 | $(16*16-15)^3 = 241^3$ | 13,997,521 | L L L L F F D D | 00 | 10 | 01 | 10 |
| **6 bit LCD** | | | | | | | |
| 6 | $64^3$ | 226,144 | L L L L L L | X | 00 | X | 00 |
| 7 | $(2*64-1)^3 = 127^3$ | 2,048,383 | L L L L L L F | X | 00 | 00 | 00 |
| 7 | $(2*64-1)^3 = 127^3$ | 2,048,383 | L L L L L L D | 01 | 01 | X | X |
| 8 | $(4*64-3)^3 = 253^3$ | 16,194,277 | L L L L L L F D | 00 | 01 | 00 | 00 |
| 8 | $(4*64-3)^3 = 253^3$ | 16,194,277 | L L L L L L F F | X | 00 | 01 | 00 |
| 8 | $(4*64-3)^3 = 253^3$ | 16,194,277 | L L L L L L D D | 00 | 10 | X | X |
| **8 bit LCD** | | | | | | | |
| 8 | $256^3$ | 16,777,216 | L L L L L L L L | X | 00 | X | 00 |

## 4.9 LCD Control Signals

The four LCD control signals (LCDClk, LCDDEn, LCDHSync, LCDVsync) are received from the MAC. These signals are used internal to the MOD and are also sent to the LCD. The polarity of the copies sent to the LCD is programmable as described in Table 4-14. These signals have pipeline delays added to them to compensate for delays that occur in the MOD data processing (FIFO output, Unpacker, OSD Mux, CLUT, Dither, FRM).

The LCDClk is the primary clock for the MOD. Except for the FIFO input section, all the logic in the MOD runs off of the LCDClk.

The LCDDEn (LCD Data Enable) indicates active pixels to be sent to the LCD. This signal is used to enable reads of the FIFO and allows data to flow through the MOD to the LCD.

LCDHsync (LCD Horizontal Sync) is used to reset the FIFO and indicates the start of a new line of the LCD.

LCDVsync (LCD Vertical Sync) is not used internally by MOD.

## 4.10 MOD Register Definitions.

The detailed register definitions are defined below in Table 4-14.

| Field | Addr | Bit(s) | r/w | Function |
|---|---|---|---|---|
| CLUTBypass | 003 | 4 | r/w | Color Look Up Table Bypass bit. |
| LCDVSPol | 003 | 3 | r/w | LCD Vertical Sync Polarity (active low = 0). |
| LCDHSPol | 003 | 2 | r/w | LCD Horizontal Sync Polarity (active low = 0). |
| LCDDEPol | 003 | 1 | r/w | LCD Data Enable (blanking) Polarity (active low = 0). |
| LCDClkPol | 003 | 0 | r/w | LCD Clock Polarity (use falling edge = 0). |
| FrmDiv | 007 | 4 | r/w | When set (=1) the frame counter for the FRM (frame rate modulation) algorithm increments at 1/2 the LCD frame rate. When cleared the FRM frame counter increments every frame. |
| FrmMode(1:0) | 007 | 2 | r/w | Determines how many bits of FRM (frame rate modulation) will be used.<br>FrmMode1 / FrmMode0 / Mode Selected<br>0 / 0 / 1/2 Only<br>0 / 1 / 1/2 & 1/4<br>1 / 0 / Illegal<br>1 / 1 / 1/2 & 1/3 |
| FrmPos (1:0) | 007 | 1:0 | r/w | Selects which bits of the eight bit wide R,G and B pixel codes are to be used for FRM. In the case of 1-bit FRM, the LSB of the range below is used. The bit position is selected as follows:<br>FrmPos1 / FrmPos0 / Bits Selected<br>0 / 0 / 1,0<br>0 / 1 / 1,0<br>1 / 0 / 3,2<br>1 / 1 / 4,3 |
| DitMode(1:0) | 00B | 3:2 | r/w | Selects the number of bits positions over which to dither. =0, dither is off; =1, 1-bit; =2, 2-bit; =3, 3 bit. |
| DitPos(1:0) | 00B | 1:0 | r/w | Selects the bit positions which the dither unit operates on. |
| HExpEn | 00F | 7 | r/w | Horizontal image expansion enable. When =1, the image is expanded horizontally by the amount indicated in HExpScale. In this mode, one out of every HExpScale pixels will be replicated on the screen. |

<table>
<tr><td>HExpScale(4:0)</td><td>00F</td><td>4:0</td><td>r/w</td><td>Horzontal Expand scale factor. When HExpEn =1, this value determines how often to replicate pixels. When =0, no expansion is done. When =1, every pixel is replicated. When =2, every other pixel is replicated (ie P0,P1,P1,P2,P3,P3,...). When =3, every third pixel is replicated (ie P0,P1,P2,P2,P3,...).</td></tr>
<tr><td>ShadeData</td><td>103-<br>1FF</td><td>5:0</td><td>r/w</td><td>Data to be loaded into FRM quarter intensity (1/4 and 3/4) pattern RAM at the address specified (total of 64 locations). The usage of the pattern RAM to generate grey shades is described in section 4.8.<br>The six bits of data are organized as:<br>ShadeData(5) = Red Quater Shade<br>ShadeData(4) = Red Half Shade<br>ShadeData(3) = Green Quater Shade<br>ShadeData(2) = Green Half Shade<br>ShadeData(1) = Blue Quater Shade<br>ShadeData(0) = Blue Half Shade<br><br>The address is organized as follows:<br><table><tr><td>Adrs Bit Range</td><td>Use in FRM Pattern</td></tr><tr><td>Adrs(5:4)</td><td>Frame select (0 to 3)</td></tr><tr><td>Adrs(3:2)</td><td>Row select (0 to 3)</td></tr><tr><td>Adrs(1:0)</td><td>Column select (0 to 3)</td></tr></table></td></tr>
<tr><td>FHData(1:0)</td><td>103-<br>1FF</td><td>1:0</td><td>r/w</td><td>Data to be loaded into FRM half intensity (1/2) pattern RAM at the address specified (total of 64 locations) The usage of the pattern RAM to generate grey shades is described in section 4.8.<br><table><tr><td>Adrs Bit Range</td><td>Use in FRM Pattern</td></tr><tr><td>Adrs(5:4)</td><td>Frame select (0 to 3)</td></tr><tr><td>Adrs(3:2)</td><td>Row select (0 to 3)</td></tr><tr><td>Adrs(1:0)</td><td>Column select (0 to 3)</td></tr></table></td></tr>
<tr><td>OvlData (2:0)</td><td>203-<br>25F</td><td>2:0</td><td>r/w</td><td>Overlay Palette data to be written into the Overlay Palette RAM at the adrs specified (8 locations for Red, Green, and Blue color planes). The first 8 locations are red (0 and 7 invalid), the next 8 locations are Green (0 and 7 invalid), the final 8 are Blue (0-7 invalid). Writing data to locations 0 or 7 for any color plane are undefinied (0=transparent, 7=white).. The data in this RAM becomes the 3 MSB's of the colors used for the menu, pointer, and scribble mode (palette of 512 possible colors of which 7 can be simultaneously displayed + white).</td></tr>
<tr><td>RedCLUT (7:0)</td><td>403 -<br>7FF</td><td>7:0</td><td>r/w</td><td>Red color lookup table data. This data controls gamma correction, dithering position,and "fade" presentation dissolves.</td></tr>
</table>

| | | | | |
|---|---|---|---|---|
| GrnCLUT (7:0) | 803 - BFF | 7:0 | r/w | Green color lookup table data. This data controls gamma correction, dithering position,and "fade" presentation dissolves. |
| BluCLUT (7:0) | C03 - FFF | 7:0 | r/w | Blue color lookup table data. This data controls gamma correction, dithering position,and "fade" presentation dissolves. |

# 5. MERLIN ADDRESS & CONTROL (MAC)

The following block diagram illustrates Merlin's Address and Control Block (MAC). This block forms the heart of the Merlin system and is responsible for all the major control functions such as: WRAM control, Micro Interface decode, Sync processing, Mode detection and LCD control. The following sections describe each of these blocks and there relationship with the rest of the Merlin System. Where appropriate software algorithms have been included.

In general terms the MAC Block orchestrates the movement of data into and out of the WRAM. To accomplish this function it must deal with two clock domains and two FIFO's (one on the input side and the other for output data). The FIFO's generate control signals which act as requests for WRAM resources. The MAC Block responds and in the process ensures that the input FIFO will not overflow (Except for the case of processor writes which have priority), and the output FIFO will not underflow. As a general strategy the MAC chip tries to keep the input FIFO always empty and the output FIFO always full.

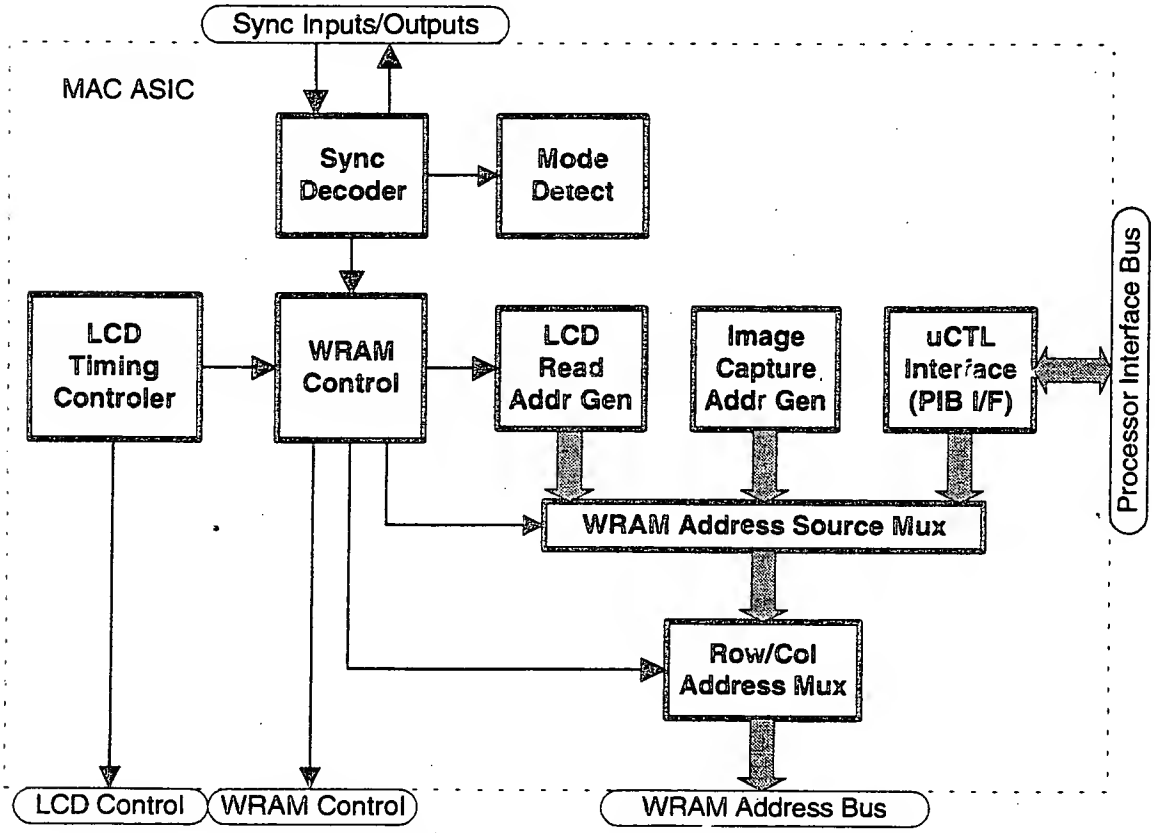



Figure 5-1: Merlin Address & Control Block Diagram.

## 5.1 MAC Register Address Map.

The software visible registers of the MAC Block are accessed through a parallel micro-controller interface which is compatible with the processor interface bus (PIB). When these registers are addressed the PIB is placed in 8-bit transfer mode. This is true even if multiple byte transfers are required to read the entire register. Software visible status registers are updated only at the beginning of VSync and a lock semaphore is provided to prevent reading data which is in the process of updating. Table 5-1 contains a list of all software visible registers in the MAC Block.

Table 5-1 MAC Register Address Map

| Base+3F0000+Offset | | | Bit Position | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Function | r/w | Offset | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Misc Status | r | 03 | MemBusy | SyncLock | | | VSP | HSP | Field | LCDVP |
| Video Status | r | 07 | NOVS | NOHS | NOCS | NOSOG | Intrlc | Comp | VPol | HPol |
| Misc Control | r/w | 0B | Lock | WakeUp | ExtVFlip | ExtHFlip | | | ProtectEn | ProtectInv |
| Capture Control | r/w | 0F | OverlayEn | MemFrz | FieldEn | IntrlcEn | DigFilter | Channel | SyncSel(1:0) | |
| Black Sample | r/w | 13 | BlkSplPol | BlkSplDly(2:0) | | | BlkSplWid(3:0) | | | |
| Block Write | r/w | 17 | BlockWr | BlockCnt(6:0) | | | | | | |
| Protect Top | r/w | 1B | ProtectTop(7:0) | | | | | | | |
| Protect Left | r/w | 1F | ProtectLeft(7:0) | | | | | | | |
| Protect Width | r/w | 23 | ProtectWidth(7:0) | | | | | | | |
| Protect Height | r/w | 27 | ProtectHeight(7:0) | | | | | | | |
| Lines/Frame | r | 2B | | | | | | LineCnt(10:8) | | |
| | | 2F | LineCnt(7:0) | | | | | | | |
| Clocks/nlines | r | 33 | ClkCnt(15:8) | | | | | | | |
| | | 37 | ClkCnt(7:0) | | | | | | | |
| ClkCnt Control | r/w | 3B | CntStart | | CntInterval(5:0) | | | | | |
| Vertical Resizing | r/w | 3F | VDrop | VRep | | VScale(4:0) | | | | |
| LCD Control | r/w | 43 | FieldLock | FieldDbl | VFlip | HFlip | LCDClr | LCDFPol | MemMode(1:0) | |
| LCD Frame Delay | r/w | 47 | LCDFrmDly(7:0) | | | | | | | |
| LCD Pulse Width | r/w | 4B | LCDVPulse(2:0) | | | LCDHPulse(4:0) | | | | |
| LCD Lines/Frame | r/w | 4F | LCDVCnt(7:0) | | | | | | | |
| LCD Clocks/Line | r/w | 53 | LCDHCnt(7:0) | | | | | | | |
| LCD Vertical Blank | r/w | 57 | LCDVBlank(7:0) | | | | | | | |
| LCD Horz. Blank | r/w | 5B | LCDHBlank(7:0) | | | | | | | |
| LCD Vert. Resolution | r/w | 5F | LCDVRes(7:0) | | | | | | | |
| LCD Horz. Resolution | r/w | 63 | LCDHRes(7:0) | | | | | | | |
| LCD Controller Base | r/w | 67 | | LCDMemBase(14:8) | | | | | | |
| Address | r/w | 6B | LCDMemBase(7:0) | | | | | | | |
| FieldControl | r/w | 6F | ExtField | FDetStart(2:0) | | | FieldInv | FDetStop(2:0) | | |
| PreScale | r/w | 73 | PreLoad | | | | | | | PreScale8 |
| PreScale | r/w | 77 | PreScale(7:0) | | | | | | | |

## 5.2 WRAM Control.

The WRAM control unit has three modes of operation and one idle state which are illustrated in the following state diagram. The highest priority mode is the display output mode used to transfer data from the WRAM core to the WRAM's SAM registers. Once started this transfer cannot be interrupted until complete. The next highest priority mode is processor read/write mode. This mode is entered when the processor is attempting to access the WRAM for menu or splash screen display. Since this mode is higher priority than image capture it is possible for the input FIFO to overflow. When this happens the MID Block asserts InFIFOempty to prevent further display writes until the start of the next line. The lowest priority mode is image capture which is used to transfer data from the input FIFO to the WRAM.

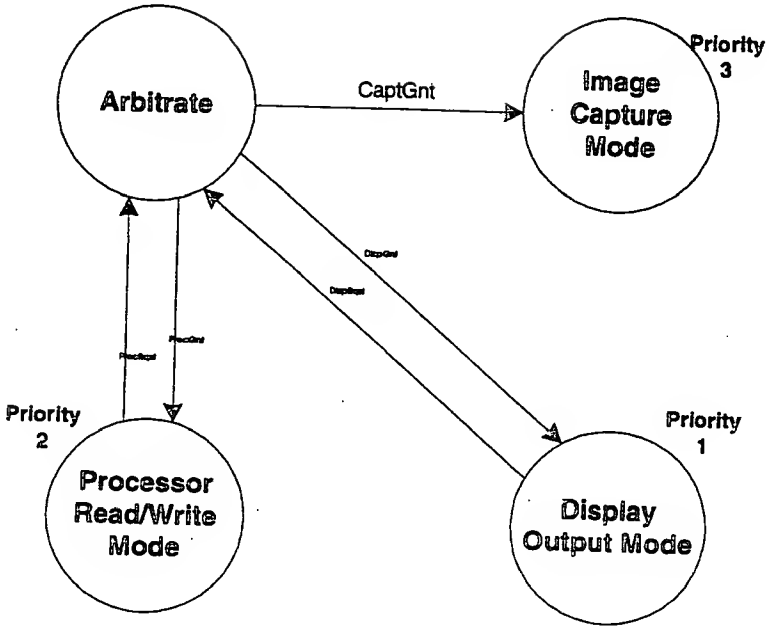



**Figure 5-2: Top Level WRAM Control State Diagram.**

5.2.1.1 Display Output Mode.

In display output mode the WRAM controller uses ultra-fast page (UFP) mode transfers to load the WRAM SAM's via the 256bit wide internal SAM transfer bus. Eight CAS cycles are required to fill the entire SAM. With the row address setup and the precharge at the end of the UFP cycle this operation takes 16 clock cycles. As indicated before once started this operation can not be interrupted.

5.2.1.2 Processor Read/Write Mode.

In processor read/write mode the WRAM controller uses a simple DRAM like read or write cycle to access the WRAM data via the parallel port. The write mask and byte enables are used to control which bits are written during write cycles. If theBlockWr bit is set then every subsequent processor write cycle will cause BlkCnt(6:0) UFP transfers with an auto increment on the address. Because of the 4/3rd's pixel packing this is only effective in clearing screen area to a monochrome shade (i.e., black,white or gery). Note that block writes work in overlay mode as well.

5.2.1.3 Image Capture Mode.

Like display output mode, image capture mode uses UFP transfers to write or read data into or out of the WRAM. In fact the row address is only loaded at the beginning of this mode and is followed by CAS cycles until a condition occurs to cause the hardware to switch out of Image Capture mode. The capture end conditions are: end of row, LCD refresh request, and processor read/write. The write mask is used to control which bits are actually written. When in this mode data is normally only written to the WRAM. However, if the DigFilter bit is set then data is read until a pixel miss occurs which then forces a switch to write mode on the following frame. Once an entire frame has been captured Merlin switches back to compare mode until the next miss is detected.

5.2.1.4 WRAM Initialization.

In order to provide for bullet proof operation Merlin reinitializes the WRAM once every frame. This will ensure that a form of transient behavior (such as power glitches) will not cause the WRAM controller and WRAM to be out of sync for more than one frame. This applies to both the input and out sides (Parallel port & SAM port) of the WRAM. This is accomplished by issuing a CBR reset cycle just following the leading edge of the first LCDHSync which occurs during the Video VSync interval. The mask register is

then automatically reloaded with the values contained in the MID Block (This is required because these registers are reset by the CBR cycle).

5.2.1.5 WRAM Refresh.

Merlin issues a refresh cycle (two cycles are issued for interlaced modes) to the WRAM once every LCDHSync. Since all Merlin compatible LCD's have more than 512line and run at >60Hz refresh rate this method of issuing refresh will more than satisfy the 512 rows in 17ms WRAM requirement.

### 5.2.2 WRAM Address Generation.

The WRAM address is a nine bit multiplexed row/column address. This address is used for both parallel port and SAM transfers. The three address sources in order of priority are: LCD Read, micro Interface, and Image Capture. A 3:1 mux-register is used to select the current source. This is followed by a row/column address mux-register which selects the source for the WRAM address bus.

5.2.2.1 Image Capture Address Generation.

The image capture address generator is responsible for controlling the address used for writing video input data into the WRAM. The algorithm used for sequencing the write address changes based on the state of the SVGAEn bit. If the SVGAEn bit is cleared (VGA Mode) then a single WRAM bank is used where each row is 512 words long. In this mode the input write address is incremented once every word transfer and advanced to the next line when the LineAdvance signals transitions from a zero to one. Otherwise, when SVGAEn=1 a two bank memory architecture is used. In this case the write address increments once every two WRAM write cycles. Note that the address is not allowed to increment into the next row or wrap from the end of display memory back to the top. When line dropping (VDrop=1) the LineAdvance signal will be ignored every VScale lines.

5.2.2.2 Row/Column Address Mux/Reg.

The row column address mux register selects the source for the nine bit WRAM address bus. The row address is selected at the start of UFP transfers followed by the column address which u... n sources the address until the transfer completes.

### 5.2.3 WRAM Timing.

All of Merlin's WRAM interface signals are referenced to the rising edge of the system clock with the exclusion of nCAS which is an inverted & gated version of SysClk. A key point to the high operating frequency is that nCAS through magic inside the MAC Block is aligned to within 2ns of SysClk. In fact the MAC Block guaranties that nCAS slightly proceeds SysClk. This is accomplished by using the MAC as the clock distribution network for SysClk. The following block diagram (Figure 5-3) illustrates the distribution method.

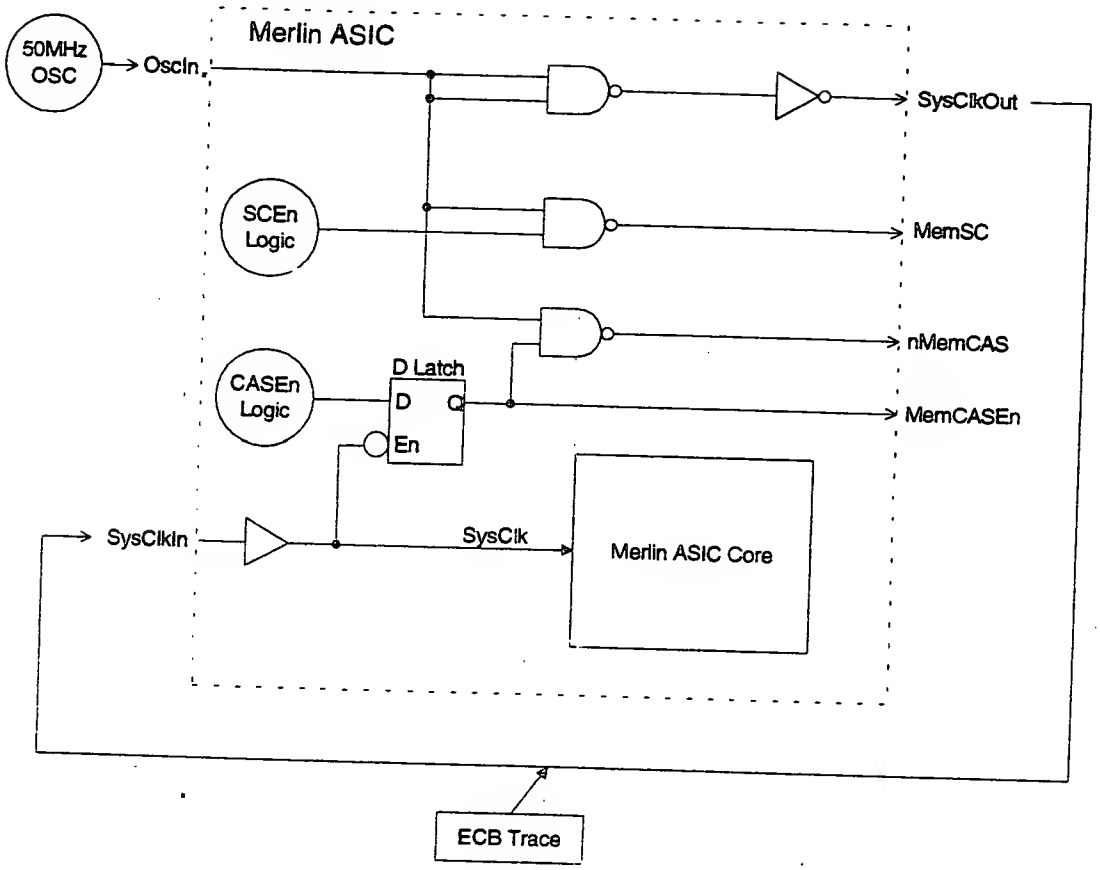



**Figure 5-3: nCAS & SysClk Distribution Method.**

The following timing diagram illustrates a basic ultra-fast page (UFP) mode transfer which is the most commonly used memory cycle in the Merlin architecture. The UFP memory operation is used both for input data write cycles and output SAM transfer cycles. The timing diagram assumes a worst case clock to out delay on the ASIC of 7ns from pin to pin.

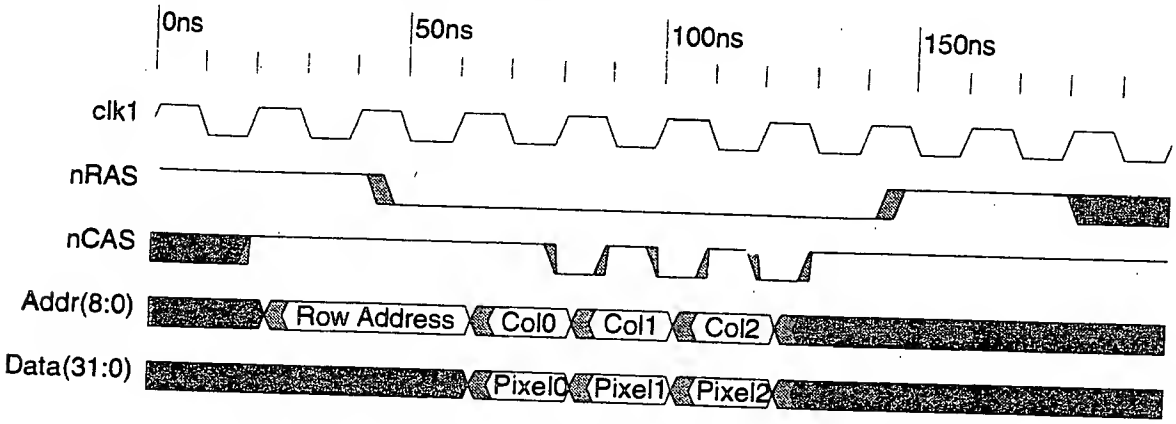



**Figure 5-4: Basic UFP WRAM Timing.**

## 5.3 LCD Control.

The MAC Block provides a general purpose LCD control unit which allows software configurable timing of the LCD. Only progressive scan (i.e., not dual scan) LCD's are supported. Under software control the vertical and horizontal intervals can be specified to within eight lines or clocks respectively. The vertical and horizontal blanking can be determined to within one line or clock. In addition MAC supports an active region window which allows firmware to specify the size and position of active data on the LCD. This allows for seamless support of resolutions which are less than the display resolution. The following diagram illustrates how the contents of the frame memory is mapped onto the LCD display.

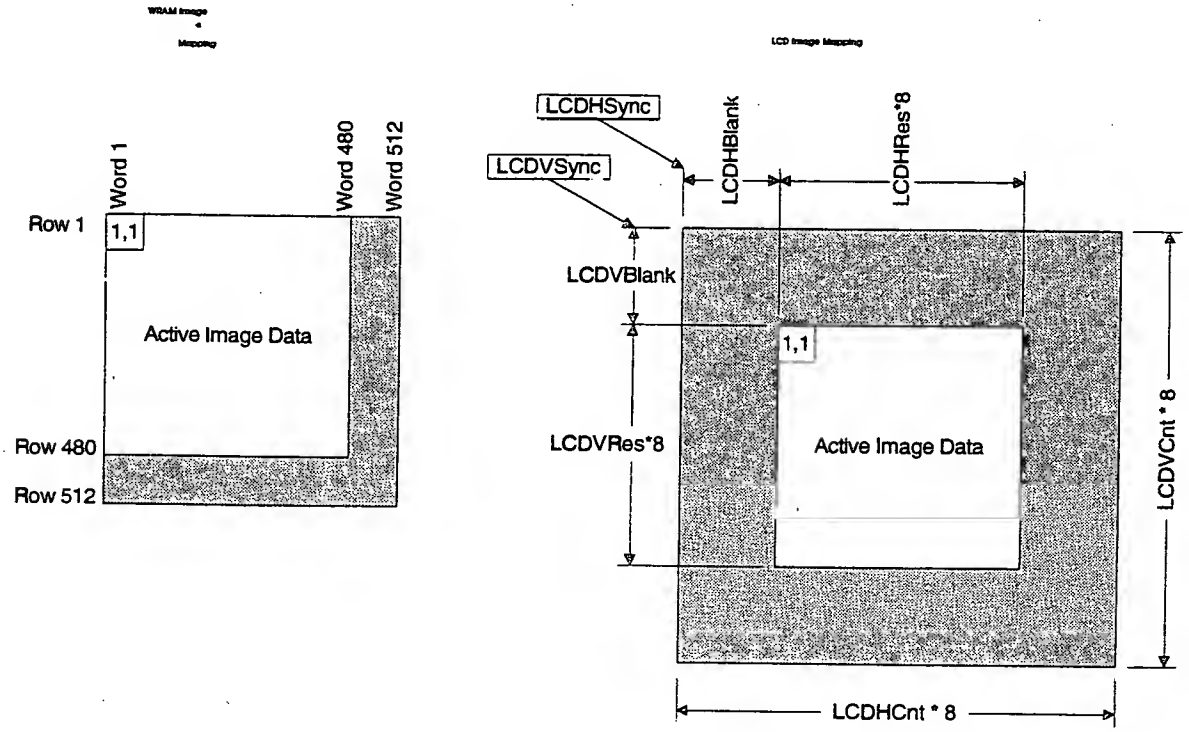



**Figure 5-5: Display Memory to LCD Image Mapping (640x480 Case).**

### 5.3.1 LCD Timing Requirements.

For detailed timing requirements of TFT LCD's refer to a manufacture's specification. In general terms TFT's require a vertical sync (LCDVSync), horizontal sync (LCDHSync), data enable (LCDDEn), clock (LCDClk) and digital RGB data whose bit width depends on the LCD type. Merlin supports 3,4,6 and 8

bit LCD's. The following two figures illustrate in terms of timing diagrams the pictorial information of

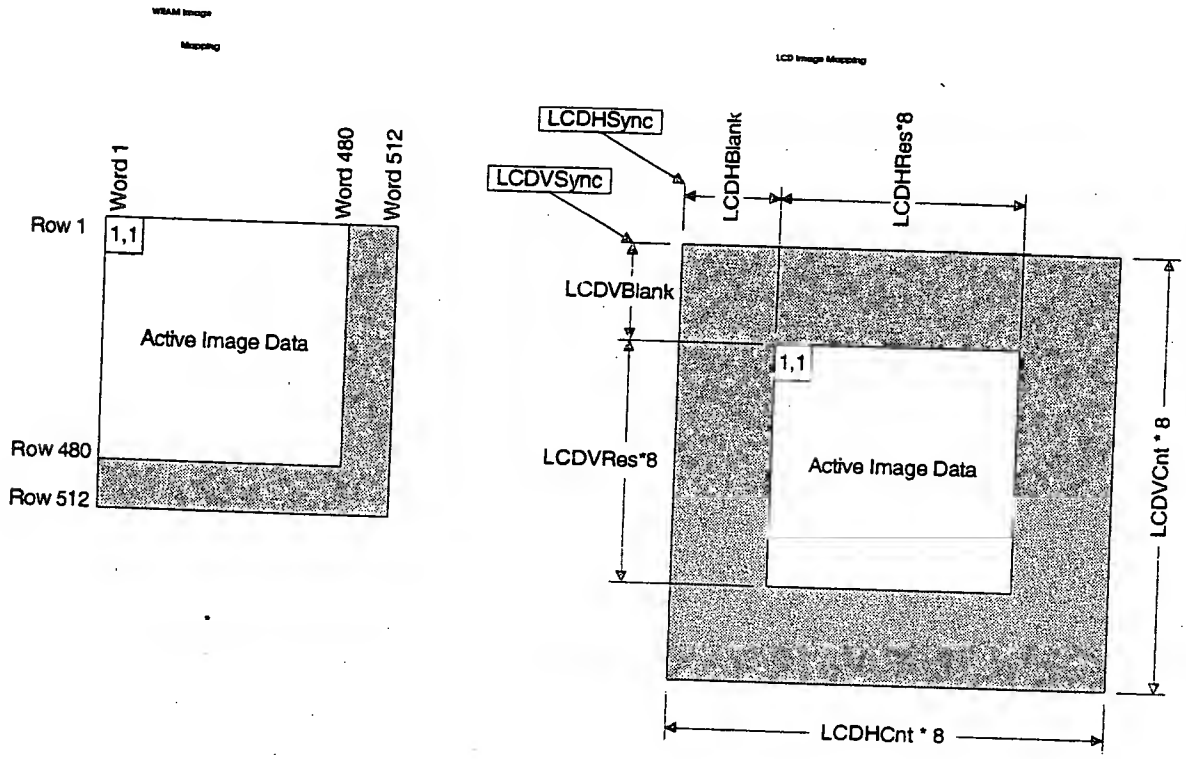



Figure 5-5. Note that the LCDActEn signal is an internal signal which indicates the start of active data. This differs from the data enable signal LCDDEn which is true when data is sent to the LCD. For resolutions which are equal or lager than 640x480 the LCDDEn and LCDActEn signals will have the same timing.

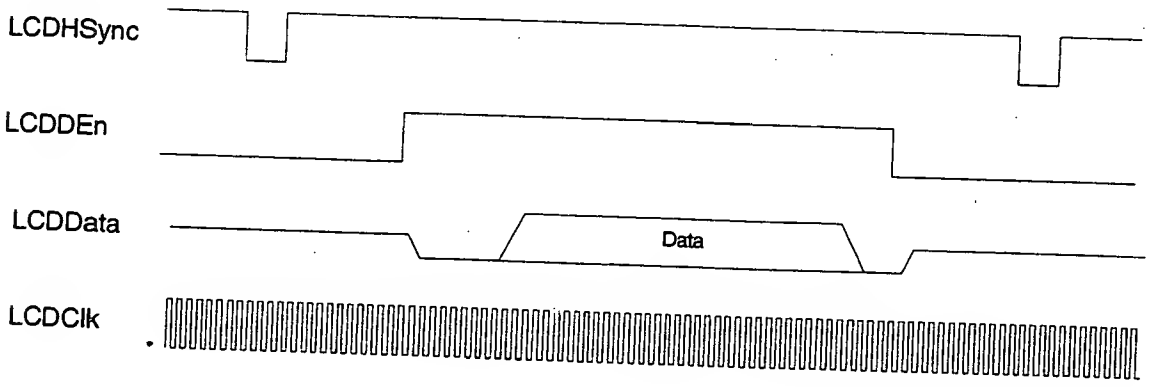



**Figure 5-6: LCD Horizontal Interval Timing Diagram.**

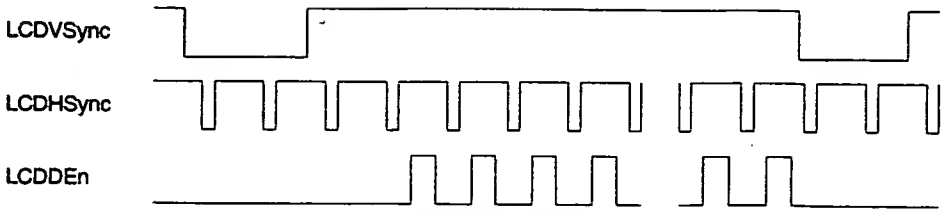



**Figure 5-7: LCD Vertical Interval Timing Diagram.**

5.3.1.1 Frame Locked Refresh Mode

When the FieldLock bit is set in the LCD control register the system synchronizes the LCD output and video input sections. This is accomplished by using the video's vertical sync pulse to start LCD frames. The firmware must configure LCDHCnt such that LCD lines per frame spec is not violated. For instance, on the Sharp 021 panel the minimum lines per frame is 516 and the maximum is 560 which implies that for a 25Mhz LCD clock and a 60Hz refresh rate the LCDHCnt register must be set as follows:

$$LCDHCnt < Min((25*10^6)/(60*516),900) = 807$$

$$LCDHCnt > Max((25*10^6)/(60*560),770) = 770$$

Note that the min and max functions are required to prevent violating the LCD clocks per line spec. In order to prevent frame splicing the LCD VSync signal must be delayed from the Video VSync by enough lines to ensure that the LCD refresh reads do not catch up with video writes. This guaranties that data from only one field is ever present on the display at any given time. The frame delay is specified in the LCDFrmDly register which is set by firmware as follows:

Assume:

VSync Rate = 60Hz.
Horizontal Rate = 31.47KHz.
Video Vertical Blanking = 33 Lines.
Video Lines Per Frame = 525.
LCD Clock Rate = 25Mhz.
LCD VBlank = 34.
LCDHCnt = 770 (see above calculation).

Then:

$$LCDFrmDly > (((480+33)*(25*10^6))/(770*31.47*10^3))-(480+34) > 16$$
$$LCDFrmDly < (((525+33)*(25*10^6))/(770*31.47*10^3))-(480+34) < 61$$

5.3.1.2 Field Doubling Refresh Mode

For cheap de-interlacing Merlin supports doubling of each field of information. This mode only operates when in frame lock refresh mode. To enable field doubling the firmware must set the FieldLock and FieldDbl bits. In this mode the LCD address generator line doubles each field, prcessing the even lines on one field and the odd lines on the next field. This mode only affects the output side, the input address and data path operate as normal with both fields of information stored in the frame memory. When field doubling the data is mapped onto the display as follows:

| Display Row | Even Frame | Odd Frame |
|---|---|---|
| 0 | Line 0 | Black |
| 1 | Line 0 | Line 1 |
| 2 | Line 2 | Line 1 |

| | | |
|---|---|---|
| 3 | Line 2 | Line 3 |
| 4 | Line 4 | Line 3 |
| 5 | Line 4 | Line 5 |
| 6 | Line 6 | Line 5 |
| etc. | ... | ... |

**Figure 5-8: Line doubled display mapping.**

### 5.3.2 LCD Read Address Generation (Display Output Mode).

The LCD image data is read from the WRAM, as was described earlier, by clocking data into the SAM holding register via UFP transfers and then scanning the data out sequentially over the sixteen bit SAM output bus. The LCD read address controller supports a number of different addressing modes required to support the rear-project (east-west flip) and ceiling mount (north-south) features of Merlin based projection products. This leads to the four possible addressing modes illustrated in Figure 5-9. The first is "Normal Project" mode, a simple sequential scan addressing scheme which scans from left to right and top to bottom (same as a CRT). Next is the "Vertical Flip" mode where lines are scanned left to right starting at the bottom of display memory and finishing at the top resulting in a vertically flipped image. This mode is only needed when products are mounted upside down. For "Rear Project" mode data is scanned right to left and top to bottom, resulting in a horizontal flip of the image. Finally the "Ceiling Mount" mode combines the previous two resulting in a right to left and bottom to top addressing mode. This causes the image to flip vertically and horizontally.

| | **Normal Project**<br>VFlip=0<br>XVFlip=X<br>HFlip=0<br>XHFlip=X | **Vertical Flip**<br>VFlip=1<br>XVFlip=0<br>HFlip=0<br>XHFlip=X | **Rear Project**<br>VFlip=0<br>XVFlip=X<br>HFlip=1<br>XHFlip=0 | **Ceiling Mount**<br>VFlip=1<br>XVFlip=0<br>HFlip=1<br>XHFlip=0 |
|---|---|---|---|---|
| Row1 | 00000 | 3BE00 | 001D8 | 3BFD8 |
| | 00008 | 3BE08 | 001D0 | 3BFD0 |
| | 00010 | 3BE10 | 001C8 | 3BFC8 |
| | ... | ... | ... | ... |
| | 001D8 | 3BFD8 | 00000 | 3BE00 |
| Row2 Start | 00200 | 3BC00 | 003D8 | 3BDD8 |
| | 00208 | 3BC08 | 003D0 | 3BDD0 |
| | ... | ... | ... | ... |
| Last Address of Row 480 | 3BFD8 | 001D8 | 3BE00 | 00000 |

**Figure 5-9: LCD Read Address Sequencing Example (640x480).**

One very important point involving details of the WRAM SAM is that for the two rear project modes data will be ordered in packets of eight words which must be reversed in the output data path block (MOD). This requirement is caused by the hardwired 256 bit wide (8 column) SAM transfer bus internal to the

WRAM. There is no way of internally reordering the data, leading to the external circuit requirement. In addition the unpacker operation is changed for these modes. Table 5-2 illustrates the equations used to calculate the LCD read address for 640x480 LCD's. It is left to the reader to extend this to the 800x600 case.

| | |
|---|---|
| Normal Project | Address = 0x200*(row) + col8; |
| Ceiling Mount | Address = 0x3BE00 - 0x200*(row) + col8; |
| Rear Project | Address = 0x200*(row) + 0x1D8 - col8 |
| Ceiling Mount/Rear Project | Address = 0x3BFD8 - 0x200*(row) + col8; |
| Note: col8 is an increment by 8 column counter. | |

This section is also responsible for Vertical image expansion by selective replicating lines of data to be sent to the LCD. This feature is controlled by the VExp =1, and the VScale register. VScale determines how often to replicate a line (1 out of every VScale lines is replicated). Vertical compression is done on the input side of MAC, horizontal compression is done in MID, and horizontal expansion is done in MOD.

## 5.4 Sync Decoder.

Video signals utilize horizontal and vertical sync signals to indicate the start of lines and frames. These signals come in many flavors with no real cross platform standards in place. Even the VESA standards developed for PC's are often violated by the various video card manufactures, and Apple has no less than three different composite sync formats! Add to this a plethora of third party video products for amplifying and distributing video signals results in one large compatibility issue. The purpose of the Sync Decoder is to convert the majority of these sync inputs into one common format which the rest of the Merlin system can rely upon. The robustness of this logic is key to the overall compatibility of the Merlin system.

### 5.4.1 Sync Detection & Selection

The Sync Decoder has two channels of sync input each of which have four types of video sync information: VSync, HSync, CSync, and SOG (sync on green). The Channel control bit in the Misc. Control register selects which of the two sources is input to the decoder. The four status bits which indicate the activity of decoders input are: NOVS, NOHS, NOCS, and NOSOG. The NOVS bit indicates that the VSync rate is less than 40Hz. The NOHS, NOCS and NOSOG status bits indicate that these inputs have a frequency of less than 10Khz. Based on these status inputs the firmware sets the SyncSel control bits (see following table) thereby determining which sync lines the sync decoder will process.

Table 5-3 Sync Decoder Sync Source Selection

| Video Status | | | | Sync Source | Sync Decoder Input | |
|---|---|---|---|---|---|---|
| NOVS | NOHS | NOCS | NOSOG | SyncSel(1:0) | VSIn | HSIn |
| X | 1 | 1 | 1 | 00 | X | X |
| X | X | X | 0 | 10 | X | SOG |
| 0 | 0 | X | X | 00 | VSync | HSync |
| 0 | 1 | 0 | X | 01 | 0 | CSync |
| 1 | 0 | 1 | 1 | 00 | 0 | HSync |
| 1 | X | 0 | X | 01 | 0 | CSync |

The no video condition is indicated when NOVS=X (Don't care), NOHS=1, NOCS=1 and NOSOG=1. For non-composite signals energy star status can be achieved by following the VESA DPMS power control strategy shown in the following table. Note that since "Suspend" mode and composite sync on HSync can not be distinguished the firmware shall assume a composite sync input is present and disable VESA DPMS until such time as NOVS becomes false. The presence of VSync's indicates that a non-composite signal is present. The firmware shall also provide a user controllable VESA DPMS enable switch. The burden of managing the power saving is placed on the firmware (i.e., firmware must tum the light source on & off depending on the VESA DPMS state).

| Video State | NOVS | NOHS | Video Image | Compliance Requirement | Power Saving | Recovery Time |
|---|---|---|---|---|---|---|
| On | 0 | 0 | Active | Mandatory | None | N/A |
| Stand-by | 0 | 1 | Blanked | Optional | Minimal | Short |
| Suspend | 1 | 0 | Blanked | Mandatory | Substantial | Longer |
| Off | 1 | 1 | Blanked | Mandatory | Maximum | System Dependent |

### 5.4.2 Sync Decoding

The sync decoder processes the VSIn and HSIn signals produced by the sync selection unit. The primary function of the decoder is to determine the type of sync input and decode it into a standard separate VSync & HSync with a positive polarity and a single video clock pulse width. In order to accomplish this the decoder must polarity correct the inputs and separate horizontal and vertical sync information from composite sync inputs if they are present. If a composite signal input is detected on HSIn the decoder sets the Comp status bit and strips the VSync pulse of off the composite input. A polarity switch of HSIn is used to indicated the presence of composite type inputs. The VPol and HPol sync polarity status bits are produced by the sync decoder and are valid only for non-composite sources. Both active high and low composite signals are allowed with full support for all of the composite types detailed in the next section. The decoder must be able to strip at least sixteen horizontal serration pulses prior to loosing lock. The worst case sync acquisition time must be no more than two frames. The sync acquisition time is defined as the time the decoder takes to lock on to the horizontal sync timing such that the serration's can be striped. This circuit can be implemented with a digital PLL which divides the horizontal interval into eight equal regions.

In addition to sync decoding this circuit produces the Field status bit by looking at the when VSync transitions occur in the horizontal interval. VSync's which occur in the middle of the horizontal interval (+/- 1/4 interval) are odd fields, otherwise, an even field is indicated. Another possible technique to determine the odd/even field is to count the number of lines per frame which varies by one on interlaced signals.

#### 5.4.2.1 Composite Sync Timing.

Sketched in Figure 5-10 are some common types of composite sync signals. Note the polarity inversion during the vertical interval which is common to all composite sync signals. This can be used to detect the presence of composite signals (i.e., the Comp status bit could be derived from this) and to enable the detection of VSync.

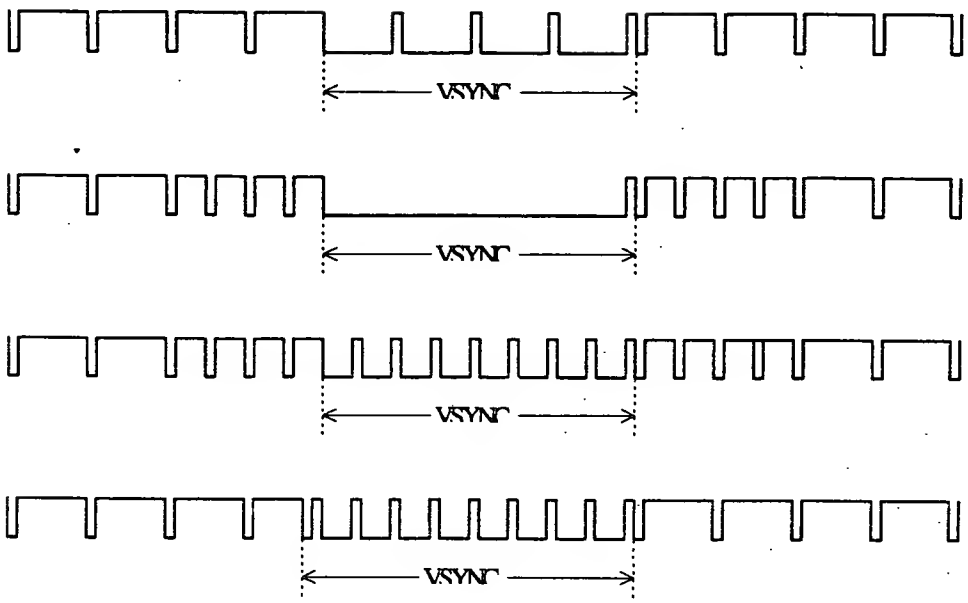



**Figure 5-10: Composite Sync signal examples.**

## 5.5 Mode Detection.

Mode detection is accomplished by timing the horizontal and vertical interval of the input source. Two software readable register, LineCnt & ClkCnt, contain information which reflects the timing for the previous vertical interval.

The line counter (LineCnt) is 11 bits in length and reflects the number of lines in the vertical interval. For the interlaced video consecutive odd and even fields are use to determine LineCnt (Note: If for interlaced signals consecutive fields are both even or odd then the LineCnt register is not updated).

The horizontal interval is determined by counting a fixed frequency clock (SysClk) for the number of lines specified by the clock count interval register (CntInterval). The 16 bit register ClkCnt is updated once every field with the new SysClk count value. The value of the CntInterval register should be set to maximize the ClkCnt register for the longest supported horizontal interval. For instance assuming a SysClk frequency of 50Mhz and a min HSync frequency (max interval) of 15.98khz (NTSC) then the following calculation illustrates how CntInterval can be computed.

Equation for computing value of clock count register:

$ClkCnt = CntInterval * (50x10^6/15.98x10^3)$

$= CntInterval * (3130)$

To prevent overflow ClkCnt must be less than 65536 ($2^{16}$):

$CntInterval * (3130) < 65536$

$CntInterval < (65536) / (3130)$

$CntInterval < 20.9$

Therefore a value of 20 would maximize the ClkCnt register without possibility of overflow.

Normally the CntInterval register would be set at power up and remain unchanged while the unit is in operation. However, CntInterval can be used to resolve two extremely close modes by loading it with a

value optimized for the particular frequency in question. With currently known modes this two phase approach is not required as long as a 50ppm crystal is used for generation of SysClk.

#### 5.5.1.1 Mode Detect Algorithm

Mode detection is accomplished by determining if the ClkCnt, LineCnt and video status registers have changed since the last mode was set. If they have then the new video mode must be determined by searching a table for the nearest match. If no sufficiently close match can be found then a new mode is created using the nearest match as a template. The following algorithm can be used as a template for the new mode detect core. If a Kernel is used then a more process oriented algorithm could be created. Note that the auto sync and auto tracking algorithms inferred in this algorithm can be found in the auto clock phase recovery discussion contained in this document.

```
main () {
        .....
        /* Somewhere in the background loop */
        /* read current state of the mode detection hardware */
        rClkCnt = ReadReg(MAC,ClkCnt);
        rLineCnt = ReadReg(MAC,LineCnt);
        rVideoStatus = ReadReg(MAC,VidStat);
        /* see if video mode has changed */
        if( (abs(gClkCnt - rClkCnt) > 6) ||
            (abs(gLineCnt - rLineCnt) > 2) ||
            (gVideoStatus != rVideoStatus)) {
                /* Turn on the blank LCD bit in the MAC Block */
                BlankLCD();
                /* wait for ClkCnt and LineCnt register to stabilize returns a FALSE condition if a timeout
                occurs (i.e. video never becomes stable. */
                if(WaitForStableVideo()) {
                        /* Determine the closest video mode */
                        videoMode = SearchModeTable();
                        /* see if the nearest video mode has significantly different timing implying a new
                        video mode has been encountered */
                        if(NewVideoTiming()) {
                                /* Add new mode to video mode table using the nearest mode found as a
                                template. */
                                AddVideoMode();
                        }
                        /* Configure hardware for the new video mode */
                        SetVideoMode();
                        /* Save video timing parameters for monitoring new mode */
                        gClkCnt = rClkCnt;
                        gLineCnt = rLineCnt;
                        gVideoStatus = rVideoStatus;
                        /* Try and find good phase and tracking setting */
                        while(!AutoPhase()) {
                                if(timeout condition) break;
                                AutoTrack();
                        } /* end while */
                } else {
                        /* Video mode is unstable set the no video condition and prompt the user with a
                        unstable video error message, suggest checking cable connections. */
                        SetNoVideo();
                } /* end if */
        } /*end if */
} /* end main */
```

## 5.6 Black Sample Control.

For black level DC restore the MAC Block produces a black sample pulse (BlackSample) which is used by the analog front end to set the black level close to zero volts. For fine black level control the MOD Block has a holding register which is clocked with the current black level on the trailing edge of BlackSample. This value can be used by firmware to individually tune the reference level of each ADC. BlackSample starts BlkSplDly clock's after the trailing edge of HSync and the pulse width is determined by BlkSplWid(3:0) times eight. BlackSample is referenced to the fixed frequency WRAM clock (SysClk) making configuration of this register independent of the video mode. The pulse width requirement of BlackSample is determined by the requirements of the analog front ends DC restore circuit.

## 5.7 Micro Controller Interface.

The micro interface consists of: 32 bits of multiplexed address/data, and various control signals. 32 bit transfers are used only for display memory access, and all register operations are limited to an eight bit bus width. Durring the address portion of a bus cycle the most significant bit is a global Merlin chip enable. The bit is generated by address decode on the processor module. The MAC Block is responsible for controlling the Merlin side of the micro controller interface. This includes data acknowledge logic and generating chip selects for the MID and MOD Block's. Display memory write cycles are posted into a holding register allowing for fast access. If the holding register is full then the acknoledge signal is with

held until write can complete. Merlin attempts to use UFP transfers on consecutive requests when they map to the same WRAM row and the transfer has not been interrupt by display or capture cycles. Note when processor writes occur capture cycles are treminated until the start of the next line.

### 5.7.1 Address Decode.

The MAC Block decodes the high order address bits into four chip selects. One for each of the three block's and another for display memory space. The following table shows the address to chip select mapping.

Table 5.5 Address Space Decode CS Mapping

| Address Offset | Function Selected |
|---|---|
| 0x000000-0x3CFFFF | Display Memory |
| 0x3D0000-0x3DFFFF | MID Block |
| 0x3E0000-0x3EFFFF | MOD Block |
| 0x3F0000-0x3FFFFF | MAC Block |

### 5.7.2 On Screen Display (Overlay's).

Merlin uses the write mask of the WRAM to allow for the least significant bit of each byte in a word to be protected for screen overlays. These bits are then used to select an overlay color or "transparent". One key issue is how is the system place into overlay mode without the screen being corrupted with a random display of overlay data. This is an issue because prior to entering overlay mode the least significant bit of display memory contains pixel data from the captured image. As a result care must be taken in activating the overlay mode as shown in the following algorithm.

1. Set MemFrz = 1 {Freeze memory until write mask is set.}
2. Set WriteMask = 0xFE {Enables write to LSB only.}
3. Clear all of WRAM using block writes (<10ms).
4. Wait For MemBusy to Clear
5. Set MemFrz = 0 {Unfreez memory.}
6. Set OverlayEn = 1 {Enable overlays.}
7. {Now you can draw you menu's & overlay's.}

## 5.8 MAC Register Definitions

Table 5-1 MAC Register Definitions

| Field | Addr | Bit(s) | r/w | Function |
|---|---|---|---|---|
| MemBusy | 03 | 7 | r | MemBusy indicates if the Processor to memory interface is busy. This bit should be used to prevent bus timeouts when performing block writes. Upto three block writes may be posted before MemBusy must be monitored. |
| SyncLock | 03 | 6 | r | Status bit indicating that the sync decoder is locked. This bit will toggle on and off for video sources which have serations. |
| VSP | 03 | 3 | r | Set by leading edge of VSync and cleared when writes to display memory start (i.e., at the end of vertical blanking). |
| HSP | 03 | 2 | r | Set by leading edge of HSync and cleared when writes start to display memory (i.e., at the end of horizontal blanking). |
| Field | 03 | 1 | r | ODD/EVEN Field indicator for interlaced video sources. Set to one on odd fields and cleared on even fields. |
| LCDVP | 03 | 0 | r | Set at the start of the LCD's vertical sync pulse and cleared by the first LCD horizontal sync which contains data (i.e., at the end of vertical blanking). |
| NOVS | 07 | 7 | r | No VSync detected status bit. Generated via time-out on vertical sync input. This signal conforms to the VESA DPMS standard (i.e., NOVS=1 when VSync frequency is less than 40Hz. For composite signal sources this bit has no meaning. |
| NOHS | 07 | 6 | r | No HSync detected status bit. Generated via time-out on horizontal sync input. This signal conforms to the VESA DPMS standard (i.e., NOHS=1 when HSync frequency is less than 10KHz. Note that the MODSRC determines which of the two sources is used to determine this status. This status bit reflects the state of the currently selected HSync source as follows: |

| | | | | |
|---|---|---|---|---|
| | | | | SyncSel(1:0) / HSync Source<br>00 — HS pin<br>01 — SOG pin<br>10 — CS pin |
| NOCS | 07 | 5 | r | NOCS=0 (False) when the CS input pin has a pulse frequency greater than 10KHz. |
| NOSOG | 07 | 4 | r | NOSOG=0 (False) when the SOG input pin has a pulse frequency greater than 10HKz. |
| Intrlc | 07 | 3 | r | Intrlc=1 when a interlaced signal source is detected. |
| Comp | 07 | 2 | r | Comp=1 when a composite sync signal source is detected. Like the NOHS status bit this bit reflects the state of the currently select horizontal sync source. |
| VPol | 07 | 1 | r | Vertical sync polarity, set to one when positive polarity VSync is detected. |
| HPol | 07 | 0 | r | Horizontal sync polarity, set to one when positive polarity HSync is detected. |
| Lock | 0B | 7 | r/w | When the lock bit is set no updating of the mode detect registers will occur. This allows the software to get an instant snap shot of the current mode detect counts. This register bit should be set prior to reading LineCnt or ClkCnt and cleared once both registers have been read. |
| WakeUp | 0B | 6 | r/w | This bit must be set 250us after power on. Seting this bit causes the WRAM controller to initialize. |
| ExtVFlip | 0B | 5 | | External vertical flip enable. |
| ExtHFlip | 0B | 4 | | External horizontal flip enable. |
| ProtectEn | 0B | 1 | r/w | When set disables WRAM writes to the region specified by the protect rectangle. This allows for Adobe graphics on top of Full motion video. |
| ProtectInv | 0B | 0 | r/w | When sets inverts the protection region from inside to outside of the protection rectangle. |
| OverlayEn | 0F | 7 | r/w | Places the system is in overlay mode. |
| MemFrz | 0F | 6 | r/w | Freezes the video input section by preventing writes to display memory. Micro-controller access to display memory is unaffected. |

| | | | | |
|---|---|---|---|---|
| FieldEn | 0B | 6 | r/w | Enables the use of the field bit when in interlace mode. When this bit is not set and IntrlcEn is true then an internal field bit is generated which changes state every frame. |
| IntrlcEn | 0F | 4 | r/w | Interlace enable bit. When set (=1) the address generation unit advances two lines every line advance. The field bit determine whether odd or even lines are being written for the current frame. |
| DigFilter | 0F | 3 | r/w | Enables digital filtering of input noise. A pixel by pixel compare is used to provide a hysteresis like function. The threshold value is set in the MID Block. |
| Channel | 0F | 2 | r/w | Selects which video source is enabled. |
| SyncSel(1:0) | 0F | 1:0 | r/w | These bits determine which of the four possible Sync sources should be used as an input to the Sync decoder. See Table 5-3. |
| BlkSplPol | 13 | 7 | r/w | Specifies the polarity of the black sample pulse. When set an active low output is produced. |
| BlkSplDly(2:0) | 13 | 6:5 | r/w | Specifies the number of clocks*4 from the trailing edge of Hsync until black sample starts. (000=0,001=4,010=8, ...). |
| BlkSplWid(3:0) | 13 | 3:0 | r/w | Specifies the width of the black sample pulse (number of WRAM clocks*4). The black sample pulse starts one clock after the trailing edge of the HSync pulse. By using the WRAM clock instead of the video clock a fixed pulse width is established which is independent of the connected video source. |
| BlockWr | 17 | 7 | r/w | Specifies the number of UFP cycles to issue when doing block writes. Note bock writes only work with data whoose RGB color values are all the same (i.e., white,grey's, & black). With this bit set writing to display memory will initiate a block write off length BlockCnt(6:0). |
| BlockCnt(6:0) | 17 | 6:0 | r/w | Number of UFP transfers to issue on block writes. The Row address automaticlly increments after each write. Note block writes cannot cross WRAM row boundries (they will wrap to the begining.) |

| | | | | |
|---|---|---|---|---|
| ProtectTop(7:0) | 1B | 7:0 | r/w | Defines the top edge of a write protected region in WRAM. This register contains the number of lines*4 from the top of display memory till the protection region starts. |
| ProtectLeft(7:0) | 1F | 7:0 | r/w | Defines the top edge of a write protected region in WRAM. This register contains the number of words*4 from the left of display memory till the protection region starts. |
| ProtectWidth(7:0) | 23 | 7:0 | r/w | Defines the width of a write protected region in WRAM. This register contains the number of words*4 of the width of the protected region. |
| ProtectHeight(7:0) | 27 | 7:0 | r/w | Defines the Height of a write protected region in WRAM. This register contains the number of lines*4 of the height of the protected region. |
| LineCnt(10:0) | 2B<br>2F | 2:0<br>7:0 | r<br>r | Total number of lines in the image including blanking. This value reflects the line count over one frame for non-interlaced signals. For interlaced signals two consecutive frames are counted. |
| ClkCnt(15:0) | 33<br>37 | 7:0<br>7:0 | r<br>r | Total number of SysClk's in CntInterval number of lines. This value is computed once per frame start on the line indicated by CntStart. |
| CntStart | 3B | 7 | r/w | When true Merlin times the horizontal interval once per frame starting at line 64. Otherwise, the Clock count register is updated continuously (dangerous). |
| CntInterval | 3B | 5:0 | r/w | Clock count interval (CntInterval) determines the number of lines over which the mode detect logic times the horizontal interval utilizing the clock count register. |
| VDrop | 3F | 7 | r/w | This bit when set enables vertical line dropping which causes the input write address to ignore the line advance input once every VScale lines. |
| VRep | 3F | 6 | r/w | This bit when set enables vertical line replication which causes the output read address to stop advancing once every VScale lines. Note that in this mode the menu system will be expanded as well as the image it self. This is a limitation of this memory architecture. |

| | | | | |
|---|---|---|---|---|
| VScale(4:0) | 3F | 4:0 | r/w | These five bits specify the number of lines over which line dropping and replicating takes place (i.e., if VScale = 5 and VDrop = 1 then one out of 5 lines is dropped by freezing the input write address.) |
| FieldLock | 43 | 7 | r/w | Forces syncronization of Video and LCD refresh. |
| FieldDbl | 43 | 6 | r/w | Enables line doubling on the LCD output side. |
| VFlip | 43 | 5 | r/w | Vertical Flip (VFlip) when set causes the WRAM address unit to scan the output side row address from bottom- to-top instead of top-to-bottom, on a line by line basis. |
| HFlip | 43 | 4 | r/w | Horizontal Flip (HFlip) when set causes the WRAM address unit to scan output side column address from right-to-left instead of left-to-right. |
| LCDClr | 43 | 3 | r/w | LCD clear when set forces the LCDBlank pin true which causes the MOD Block to clear all data sent to the LCD resulting in a near instantaneous blank screen. |
| LCDFPol | 43 | 2 | r/w | LCD Field polarity, determines the polarity relationship between the Video capture side field control and the LCD output side field control when in field lock mode. |
| MemMode(1:0) | 43 | 1:0 | r/w | Sets the memory map use by Merlin.<br>MemMode(1:0) / Memory Map<br>00 / 640x480<br>01 / 800x600<br>10 / Illegal<br>11 / 800x600 1/2 Sample |
| LCDFrmDly(7:0) | 47 | 7:0 | r/w | When in FieldLock mode sets the delay in lines between the Video vertical sync and LCD vertical sync. |
| LCDVPulse(2:0) | 4B | 7:5 | r/w | Width of the LCDVSync pulse in number of LCDHSync's. |
| LCDHPulse(4:0) | 4B | 4:0 | r/w | Width of the LCDHSYNC pulse in number of LCDClk's*8. |
| LCDVCnt(10:0) | 4F | 7:0 | r/w | This register specifies the number of lines per LCD frame divided by eight. |

| | | | | |
|---|---|---|---|---|
| LCDHCnt(10:0) | 53 | 7:0 | r/w | This register specifies the number of clocks per LCD line divided by eight. |
| LCDVBlank(7:0) | 57 | 7:0 | r/w | This register specifies the number of lines following the leading edge of LCDVSync until LCD data starts. |
| LCDHBlank(7:0) | 5B | 7:0 | r/w | This register specifies the number of clocks following the leading edge of LCDHSync until LCD data starts. |
| LCDVRes(7:0) | 5F | 7:0 | r/w | This register specifies the number of lines of resolution in the display divided by eight. For a 640 line display this register would be set to $80_{10}$. |
| LCDHRes(7:0) | 63 | 7:0 | r/w | This register specifies the number of columns of resolution in the display divided by eight. |
| LCDMemBase(14:0) | 67<br>6B | 6:0<br>7:0 | r/w | LCD Memory base address. |
| ExtField | 6F | 7 | r/w | Indicates the external field input should be used instead of the internaly determined field state. |
| FDetStart(2:0) | 6F | 6:4 | r/w | Determines where in a line of video the syncdecoder starts looking for a transition on Vsync which indicates an odd field. |
| FieldInv | 6F | 3 | r/w | Inverts the sense of the detected field. Setting this bit will cause the Field status bit to invert. |
| FDetStop | 6F | 2:0 | r/w | Determines where in a line of video the syncdecoder stops looking for a transition on Vsync which indicates an odd field. |
| PreLoad | 73 | 8 | r/w | Force preload of sync decoder prescale counter. Must be set for one Hsync interval to take effect. Once loadedd this bit shpuld be cleared. This bit is only used for diagnostic purposes, and should not be set in normal operation. |
| PreScale(8:0) | 73<br>77 | 0<br>7:0 | r/w | Value to load into prescale counter when PreLoad is set. |